TARTU ÜLIKOOL
Arvutiteaduse Instituut
Informaatika õppekava

Uku Tonsiver

2D külgvaates õudusmäng “Awaken”

Bakalaureusetöö (9 EAP)

Juhendaja: Madis Vasser, PhD

Tartu 2025

2D külgvaates õudusmäng “Awaken”

Lühikokkuvõte:

Lõputöö eesmärk oli disainida ja arendada kaasahaarav 2D külgvaates õudusmäng. Selleks, et saada inspiratsiooni, teha kindlaks, mis muudab sellise mängu kaasahaaravaks, ning et luua midagi varasematest mängudest erinevat, analüüsiti varasemaid edukaid 2D külgvaates õudusmänge. Anti ülevaade arenduses kasutatud tarkvaralahendustest ning põhjendati, miks sellised valikud tehti. Kirjeldati töö käigus disainitud mängusüsteemide loogikat ja koos toimimist. Lõputöös anti ülevaade mängu põhiliste mehaanikate töötamise algoritmidest. Lisaks arendusprotsessile kirjeldati testimist ja selle tulemusi. Eraldi käsitleti alfatestimist ja beetatestimist. Testimistulemuste põhjal viidi mängu sisse muudatused.

Võtmesõnad: videomäng, mängudisain, mänguarendus, õudusmäng, Godot

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

2D side scroller horror game Awaken

Abstract:

The purpose of the thesis was to design and develop an engaging 2D side scroller horror game. In order to get inspiration, determine what makes a game in this genre captivating, and to create something unique from earlier games, earlier successful 2D side scroller horror games were analysed. An overview of software solutions used in development was provided, as well as reasonings for these choices. Game systems designed for this game and their interconnectivity were described. In the thesis, an overview of the algorithms behind the main game mechanics was provided. In addition to the development process, the testing process and its results were described. Alpha testing and beta testing were investigated separately. Based on the test results, improvements were made to the game.

Keywords: video game, game design, game development, horror game, Godot

CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

1. Sissejuhatus

Üks tuntud videomängude žanr on õudusmängud. Williamsi järgi [1] moodustasid õudusmängud 2020. aastal kõigist mängitud videomängudest 2%, samas kui õudus on Z-põlvkonna kolmas lemmik meelelahutusžanr. Sellest võib järeldada, et õudusmängude järele on suur nõudlus, kuid võrreldes muude žanritega võrdlemisi väike pakkumine, seega on see potentsiaalselt tulus žanr mänguarenduseks.

Õudusmängude žanr muutus populaarseks 1970. ja 1980. aastatel näiteks mängude "Haunted House" ja "Splatterhouse" ilmumise mõjul [2]. Tol ajal olid videomängud riistvarapiirangute tõttu valdavalt kahemõõtmelised. Alates 1992. aastal ilmunud mängust "Alone in the Dark" [3] on õudusmängude žanr olnud domineeritud 3D mängude poolt. Kuigi on loetud arv edukaid modernseid 2D külgvaates õudusmänge (neist räägitakse lähemalt teises peatükis), on see formaat 3D alternatiiviga võrreldes üsna vähe avastatud. See tekitab võimaluse luua midagi uut ja unikaalset 2D külgvaates õudusmängu žanris.

Selle lõputöö eesmärk on luua kaasahaarav 2D külgvaates õudusmäng. Arenduse ja testimise käigus saab selgeks, kui hästi sobib 2D külgvaates formaat õudusmängude jaoks ning kas sellistele mängudele leidub tulus turg.

Töö teises peatükis analüüsitakse juba olemasolevaid 2D külgvaates õudusmänge. Uuritakse kunstistiili, lugu ja mängimiskogemust. Varasemate mängude analüüsimine aitab mõista, mis muudab mängu selles žanris edukaks. Võrdlus varasemate mängudega aitab ka luua selle töö käigus uudne ja omapärane mäng.

Kolmandas peatükis kirjeldatakse arenduses kasutatud tarkvaralahendusi. Antakse ülevaade, milliseid tööriistu mis eesmärkidel kasutati ja miks sellised valikud tehti. Selgitatakse erinevate tööriistade tugevusi ja nõrkusi.

Neljas peatükk kirjeldab "Awakenit" ennast. Siin selgitatakse mängu põhilisi mehaanikaid, antakse ülevaade mängu loost, kirjeldatakse erinevaid asukohti mängus ja kasutajaliidest. Neljas peatükk annab lugejale selgema arusaama, mida "Awakenist" oodata.

Viiendas peatükis kirjeldatakse mängu arendusprotsessi. Kirjeldatakse põhjalikult ja tehniliselt alade disaini, mängusüsteemide sisemist loogikat, algoritme mängu mehaanikate taga, varjutajate kasutamist "Awakenis" ja kunsti- ning helivarade hankimist, loomist ja kohendamist. Siin selgitatakse ka, kuidas praktikas erinevaid tarkvaralahendusi ja Godoti komponente kasutati.

Kuuendas peatükis antakse ülevaade "Awakeni" testimisest. Eraldi käsitletakse alfatestimist ja beetatestimist. Kirjeldatakse erinevaid kasutatavaid testimise metoodikaid, testimise tulemusi ja muudatusi, mis nende põhjal mängule tehti.

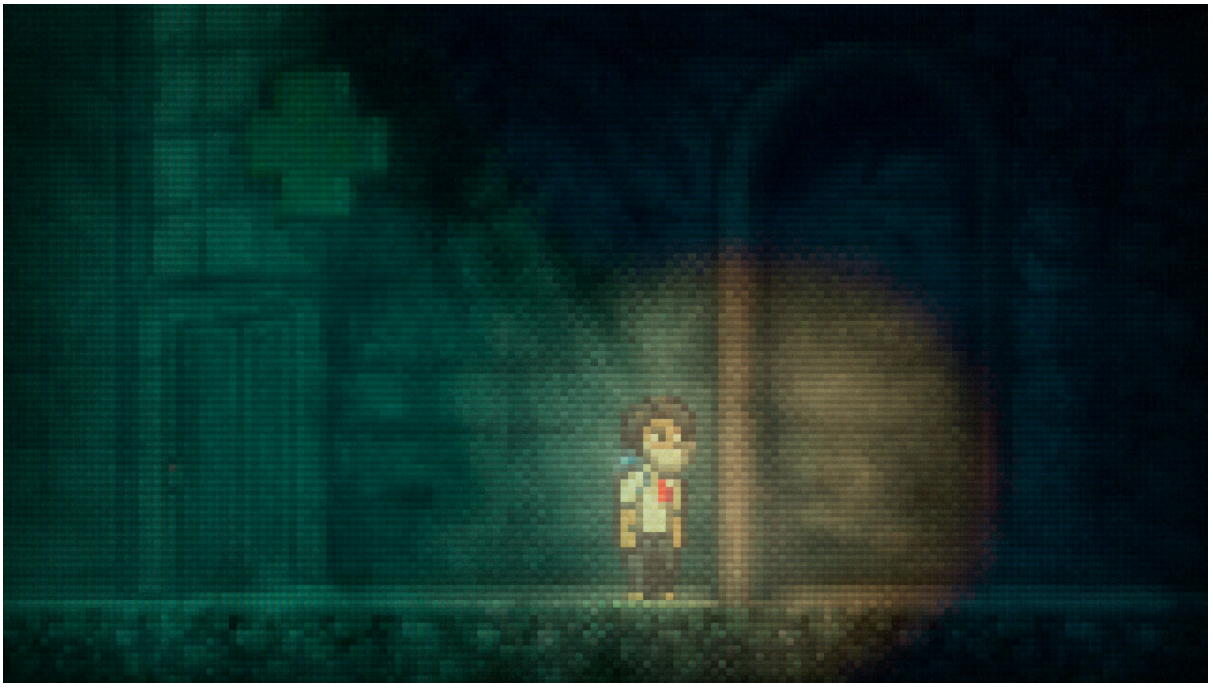
Lisa I defineerib töös kasutatavad terminid. Lisa II hõlmab lõputööle kaasnevaid faile, mis sisaldavad testimise küsimustikku ja selle vastuseid ning mängu käivitataval kujul. Lisa III sisaldab lõputöö litsentsi.

2. Sarnased mängud

Selles peatükis tutvustatakse tuntumaid olemasolevaid külgvaates õudusmänge. Olemasolevate mängude uurimine aitab leida viise, kuidas luua lõputöö käigus millegi poolest unikaalne mäng.

2.1 Lone Survivor

Järgnev lõik põhineb "Lone Survivor: The Director's Cut" müügilehel Steam keskkonnas [4]. "Lone Survivor" on 2D külgvaates psühholoogiline ellujäämise õudusmäng, kus peategelase eesmärk on põgeneda epideemiasse langenud linnast, võideldes samal ajal enda mõistuse aeglase kadumisega (vt Joonis 1). "Lone Survivor" ilmus aastal 2012 Steam keskkonnas ja 86% tema arvustustest on positiivsed.

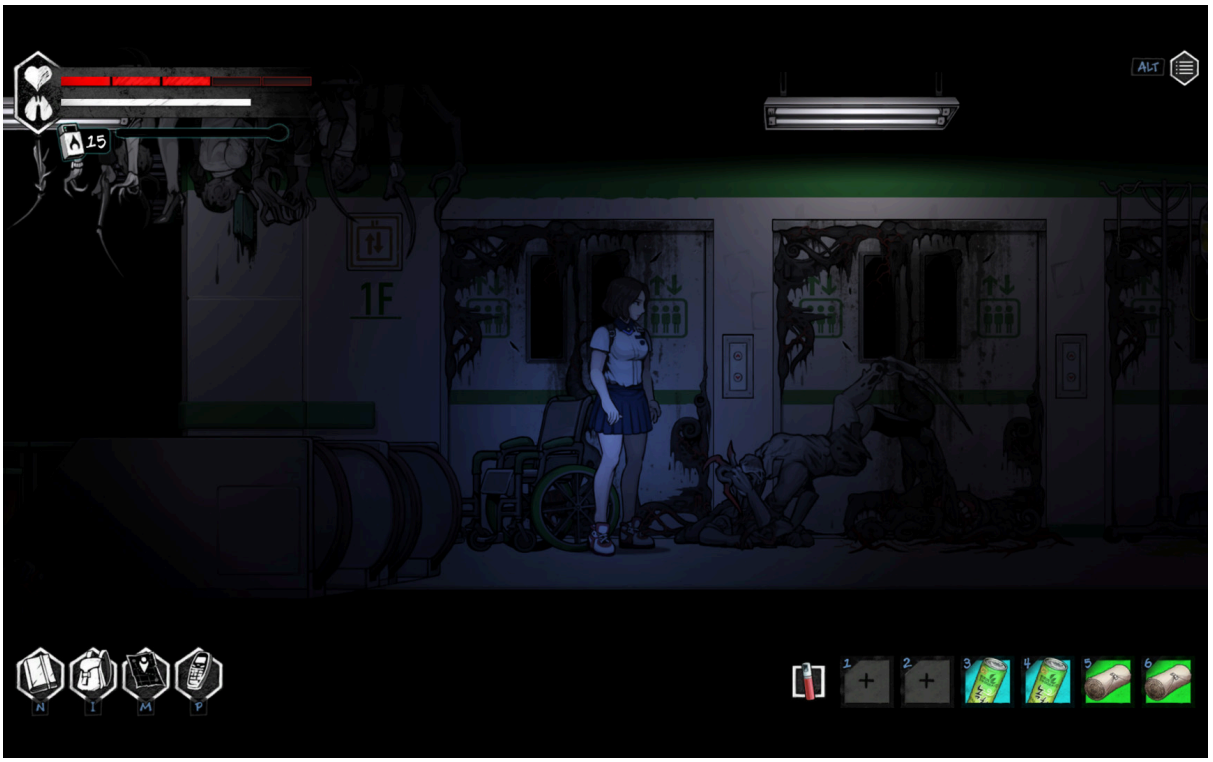


Joonis 1. Lone Survivor [4]

"Lone Survivori" kunstistiil on sarnaselt "Awakenile" *pixel art*. Erinevalt "Lone Survivorist" aga langeb "Awaken" vähem ellujäämise õuduse žanrisse ning kasutab rohkem elemente Lovecrafti õuduse žanrist. "Awakenis" on vähem rõhku võitlemisel ning enamik mängu kestusest on mängijal vaid väga piiratud võimalused ennast kaitsta.

2.2 The Coma seeria

Järgnev lõik tugineb “The Coma” mängude kollektsiooni müügilehele Steam keskkonnas [5]. “The Coma” on 2D külgvaates õudusmängu seeria, kus Sehwa High keskkooli õpilased üritavad vältida neid jahtivat olendit ning välja selgitada Sehwa High sünged saladused (vt Joonis 2). Seerias on kolm mängu: “The Coma: Cutting Class”, “The Coma 2: Vicious Sisters” ja “The Coma 2B: Catacomb”. Steam keskkonnas on mängude arvustustest positiivsed vastavalt 85%, 95% ja 79%.



Joonis 2. The Coma 2: Vicious Sisters [6]

Erinevalt “Awakenist” on “The Coma” seeria mängud *anime*-sarnase kunstistiiliga. “The Coma” mängudes on sarnaselt “Awakenile” mängija üks ülesannetest loo ja saladuste välja selgitamine mängumaailma avastamise läbi. “The Coma” mängudest suure osa moodustab vaenlase vältimine ja vaenlase eest põgenemine. “Awaken” on loo konkreetse edasi kandmise eesmärgil ja lühikese arendusperioodi tõttu lineaarsem. Kohtumised vaenlaste ja teiste tegelastega ning muud sündmused mängus on “Awakenis” suuresti kindlalt ette määratud.

2.3 Limbo

Järgnev lõik põhineb “Limbo” müügilehel Steam keskkonnas [7]. “Limbo” on 2D õuduselementidega mõistatus-platvormimäng. “Limbo” on omapärase käsitsi joonistatud

mustvalge kunstistiiliga (vt Joonis 3) ja räägib loo poisist, kes otsib oma õde metsas ja lagunevas linnas, vältides ette sattuvaid koletisi.

Joonis 3. Limbo [7]

"Limbo" loos on mitu sarnasust "Awakeni" looga. Mõlemas otsib peategelane oma õde, ja mõlemas mängus on kahtlusi, mis on päris ja mis on peategelase kujutlus. "Awaken" erineb Limbost kunstistiili ja mehaanika poolest. "Awaken" ei ole erinevalt "Limbost" platvormimäng, vaid lihtsam 2D külgvaates mäng. "Awakenis" on mõistatusmängude elemente, kuid see ei ole "Awakeni" puhul põhifookus.

3. Kasutatud tööriistad

See peatükk tutvustab mängu arenduses kasutatavaid tarkvaralahendusi. Läbiva mustrina on tehtud valikud ajendatud autori varasemast kogemusest.

3.1 Godot

Mängu arendus toimub Godot mängumootoris. Kasutusel on Godoti versioon 4.3. Godot on tasuta ja vabavaraline mängumootor, mis võimaldab nii 2D kui 3D mängude arendust [8]. Mängu skriptid kirjutatakse kasutades Godoti sisseehitatud programmeerimiskeelt GDScript, mis on süntaksilt sarnane Pythonile.

Kuna tegemist on 2D mänguga, on selles töös asjakohane just Godoti võimekus 2D videomängude loomisel. Mohdi järgi [9] on Godot tuntud võimeka mootorina 2D mängude arenduses tänu tema vähesele ressursinõudlusele, kasutajasõbralikule programmeerimiskeelele ja tasuta ning vabavaralisele loomusele. Potentsiaalse nõrkusena on Godoti noore ea tõttu kasutajakommuun alternatiivsete mängumootoritega võrreldes pigem väike, mistõttu võib abi leidmine keeruliseks osutuda [9].

Järgnev materjal tugineb Julian Holfeldi analüüsile Godoti populaarsusest iseseisvate (ingl *indie*) mänguarendajate hulgas [10]. Kuigi Godotit kasutatakse praegu mänguarenduses vähem kui alternatiivseid mootoreid, näiteks Unity ja Unreal Engine, on iseseisvate arendajate hulgas Godoti populaarsus kasvamas. Iseseisvate arendajate hulgas populaarsel veebilehel itch.io leiduvate mängude hulgas, mille mängumootor on teada, on Godot osakaal kasvanud aastate 2018 ja 2023 vahel 2,5 protsendi pealt 7,5 protsendi peale ning itch.io suurima mängujämmi, GMTK Game Jam esituste hulgas on Godot osakaal kasvanud aastate 2020 ja 2023 vahel 12,2 protsendi pealt 19 protsendi peale. Holfeld rõhutab, et need arvud on alumised piirid Godoti osakaalule iseseisvate mängude hulgas, sest erinevalt paljudest alternatiividest võivad Godoti projektid olla eksporditud mootorit tuvastavaid failinimesid kasutamata, seega kasutavad Godotit peaaegu kindlasti ka mingi osa mängudest, mille mootor on teadmata.

Käesoleva projekti mängumootoriks valiti Godot intuitiivse kasutajaliidese, vähese ressursinõudluse ja autori varasema kogemuse tõttu tarkvaraga. Kuna tegemist on tasuta ja

vabavaralise tarkvaraga, ei pea ka muretsema võimalike litsentsitasude pärast, mis alternatiividega kaasas võiks käia.

3.2 Aseprite

Kuigi enamik projektis kasutatavad kunstivarad on tasuta või tasu eest vajalike kasutus-litsentsidega internetist hangitud, kasutatakse selles projektis üksikute kunstivarade loomiseks ja mujalt hangitud kunstivarade kohandamiseks kunstitarkvara Aseprite. Kasutusel on Aseprite versioon 1.3.9.2-arm64. Aseprite on *pixel art* loomise tarkvara, millel on palju funktsionaalsusi mänguarenduses vajalike varade (näiteks animatsioonid, *tileset*id) loomise lihtsustamiseks [11].

Aseprite on tasuline tarkvara. Autor ostis Aseprite litsentsi digitaalsest levitamisteenusest Steam hinnaga 16,79€ [12]. Tasu on ühekordne ja tagab õiguse kasutada tarkvara nii isiklikel kui ärilistel eesmärkidel [11]. Ühekordne tasu annab ostjale ligipääsu ka tulevikus ilmuvatele tarkvara uuendustele [11]. Steami Aseprite müügilehel [12] on Aseprite kasutajaarvustustest 99% positiivsed.

Võimalikest kunstitarkvaradest valiti selle mängu arenduses kasutamiseks Aseprite esmalt sellepärast, et Aseprite on loodud spetsiifiliselt *pixel art* kunstivarade loomiseks ja muutmiseks, ning *pixel art* on selle mängu kunstistiil. Lisaks muutis Aseprite'i atraktiivseks valdavalt positiivsed arvustused, ühekordne tasu regulaarse tellimuse asemel ning autori varasem kogemus tarkvaraga.

3.3 Logic Pro

Sarnaselt kunstivaradele on ka enamik mängu heliefektid kas tasuta või tasuliselt vajalike litsentsidega internetist hangitud, kuid neid kohendatakse vastavalt vajadusele. Lisaks on vaja töötleda kitarril salvestatud taustamuusikat. Heliefektide ja muusika töötlemine toimub Apple helitöötlustarkvaras Logic Pro. Käesolevas projektis kasutatakse Logic Pro versiooni 10.8.

Logic Pro on tasuline tarkvara. Logic Pro on saadaval digitaalses levitamisteenuses Apple App Store ühekordse hinnaga 229,99€ [13]. Autor sai Logic Pro litsentsi ja tarkvara sülearvuti ostuga kaasa.

Kuna Logic Pro sihtgrupp on professionaalsed muusikaloojad [13], on tegemist väga võimeka ja kalli tarkvaraga. Videomängu heliefektide ja kitarrimuusika töötlemiseks pole vajadust nii keeruka tarkvara järele ning pole põhjust nii palju raha helitöötlustarkvarasse investeerida. Selles projektis kasutatakse Logic Prod varasema kogemuse pärast ja ajendatuna asjaolust, et selle litsents oli autoril juba varem olemas.

3.4 Visual Studio Code

Kuigi mängu GDScript programmeerimiskeeles kirjutatud skriptide töötlemine toimub valdavalt Godoti sisseehitatud koodiredaktoris, on muude keelte (käesolevas projektis Markdown, Bash, Python ja JSON) töötlemine autori jaoks mugavam Microsofti koodiredaktoris Visual Studio Code. Kasutatakse Visual Studio Code versiooni 1.92.2.

Järgnev materjal põhineb Visual Studio Code GitHubi repositooriumile [14]. Visual Studio Code on tasuta ja osaliselt vabavaraline tarkvaraarendustööriist. Arendust haldab Microsoft, kuid arendusse panustada saavad kõik vabatahtlikud. Visual Studio Code'il on MIT litsents, mis tähendab, et tarkvara on lubatud kasutada nii isiklikel kui ärilistel eesmärkidel ilma litsentsitasudeta.

Visual Studio Code valiti projektis kasutamiseks tasuta saadavuse ja vähese ressursinõudluse tõttu. Samuti oli valik motiveeritud autori varasemast kogemusest tarkvaraga.

3.5 Git ja GitHub

Et vältida töö kaotamist riistvara kadumise, varguse, katki minemise või inimvigade korral, ning et juhendajat tööga kursis hoida, on tähtis kasutada projektis versioonikontrolli. Selle mängu arenduses kasutatakse versioonikontrolli süsteemina Giti ning kaugrepositooriumi hoitakse Giti haldamistarkvaras GitHub.

Git on tasuta ja vabavaraline versioonikontrolli süsteem. Khleeli järgi [15] on Git üks populaarsemaid versioonikontrolli süsteeme ning pakub eeliseid alternatiivide ees, näiteks mitme panustaja paralleelse töö toetamine repositooriumi "oksadeks" organiseerimise läbi ja oksade vaheliste konfliktide lahendamise tööriistad. Suur eelis Giti kasuks on tema populaarsus, mis tagab võimalike kaaspanustajate pädevuse samas versioonikontrolli süsteemis.

GitHub on Microsofti hallatud veebirakendus Giti repositooriumide majandamiseks. GitHub on tasuta tarkvaralahendus ja pakub rea funktsionaalsusi töö lihtsustamiseks ja turvalisemaks muutmiseks.

Git ja GitHub valiti projekti versioonikontrolli lahendusteks autori varasema kogemuse ja tasuta olemuse tõttu. Mõlema populaarsusest tuleneb ka sujuv koostöö võimalike kaaspanustajatega ning juhendajaga.

3.6 Final Cut Pro

Treileri loomiseks salvestatakse mängust klippe Godoti Movie Maker tööriistaga. Klippe töötletakse ja lõigatakse kokku Apple videotöötlus tarkvara Final Cut Pro abil. Lõputöös kasutatakse Final Cut Pro versiooni 10.7.1.

Final Cut Pro on tasuline tarkvara. Apple digitaalses levitamisteenuses App Store on Final Cut Pro saadaval ühekordse tasuga 349,99€ [16]. Autor sai Final Cut Pro litsentsi kaasa sülearvuti ostuga.

Final Cut Pro on professionaalseks videotöötluseks mõeldud tööriist [16], mistõttu on tegemist üsna keeruka ja kalli tarkvaraga. Kuigi treileri töötlemisel on Final Cut Pro funktsionaalsused kasulikud, ei oleks nii suure raha investeerimine ainult treileri töötlemiseks kasutatavasse tarkvarasse mõistlik. Otstarbekam oleks kasutada tasuta tarkvara või palgata videotöötluseks professionaal. “Awakeni” treilerit töötles autor Final Cut Pro abil ise, sest Final Cut Pro kasutuslitsents oli autoril olemas juba enne lõputööga alustamist.

4. Awaken

Selles peatükis antakse ülevaade mängu sisust. Kirjeldatakse põhilisi mehaanikaid, lugu, tegelasi, mängu alasid ja kasutajaliidest.

4.1 Mehaanikad

“Awakeni” mehaanikad on üsna traditsioonilised ellujäämise õudusmängude puhul. Mängija saab maailmas liikuda, seljakotiga esemeid kaasas kanda, ennast kaitsta ja surma puhul toimub taassünd. Traditsioonilisest ellujäämise õudusmängust eristab “Awakenit” see, et võitlusmehaanika on võrdlemisi piiratud.

4.1.1 Mängija liikumine

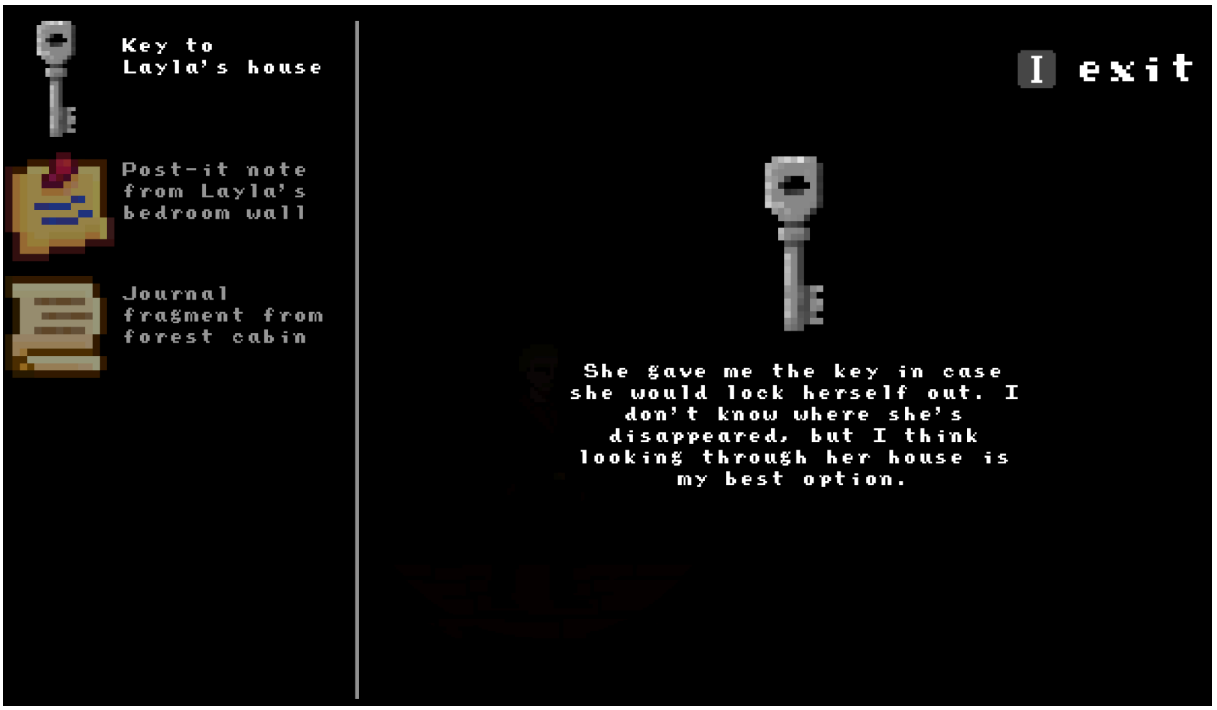
Mängija saab liikuda mööda maastikku vasakule ja paremale. Mängumaailmas on määratud hõõrdejõud, mis aeglustab sisendi lõppemisel mängijat kuni seiskamiseni. Mängijal on määratud kiirendus, mille mõjul mängija kiirendab kuni maksimaalse liikumiskiiruseni. Määratud on kõndimiskiirus ja jooksmiskiirus, millest valitakse üks maksimaalseks liikumiskiiruseks. Kõndimiskiiruse ja jooksmiskiiruse vahel saab vahetuda sprintimise sisendiga (klaviatuuril vaikimisi “Shift”, puldil “L2”). Vastavalt hetkekiirusele mängib kas kõndimis- või jooksmisanimatsioon. Alla liikumise sisendit (klaviatuuril vaikimisi “S”, puldil vasaku juhtkangi alla liigutamine) saab kasutada alla poole liikumiseks, kui on võimalus erinevatele tasemetele liikuda (vt Joonis 4).

Joonis 4. Mitme tasemeline koht

Üles liikumise võimalust ei ole. Kuigi paljudes külgvaates 2D mängudes väljendub üles liikumine hüppamisena, tehti otsus see "Awakenist" välja jätta. Kuna mängus on sujuv maastik ja trepid, mitte katkendlik maastik, mida peaks hüpates ületama, ei ole hüppamiseks vajadust. Autori hinnangul ei sobiks hüppamise mehaanika ka mängu kunstistiiliga ja õhkkonnaga. Liikumise sisemist loogikat tutvustatakse peatükis 5.3.2.

4.1.2 Seljakott

Esemete kaudu loo edasi kandmiseks ja mõistatuste lahendamiseks on mängijal seljakott (ingl *inventory*). Igal esemel on seljakotis nimetus, pilt ja kirjeldus (vt Joonis 5). Esemete üles korjamine tekitab ekraanile teavituse, juhtimaks mängija tähelepanu, et seljakotti on ese lisandunud. Seljakoti sisemist loogikat tutvustatakse peatükis 5.2.2.



Joonis 5. Seljakoti kasutajaliides

Seljakoti mehaanika on tihedalt seotud võitlusmehaanikaga. Tulistamismehaanikaks on vajalik seljakoti olemasolu, et mängijakarakter saaks relva kaasas kanda.

4.1.3 Võitlus

Mängija enesekaitseks on "Awakenis" ka võitlusmehaanika. Lähedal asuvaid vaenlaseid saab mängija rusikaga lüüa. Kaugemalt on võimalik vaenlasi kahjutuks teha tulistades. Tulistada

saab ainult siis, kui mängija seljakotis on revolver, mille ta saab mängu teises pooles. Kuna tegu pole ellujäämismänguga, vaid narratiivil põhineva õudusmänguga, on tulistamine piiratud: mängijal on maksimaalselt vaid viis kuuli, mida ei saa mängu jooksul mitte kusagilt juurde. Tulistamissisendi (klaviatuuril vaikimisi "L", puldil "X") esmakordsel kasutamisel mängib kas pikk tulistamisanimatsioon, kus mängija võtab relva välja, sihib ja tulistab, või kui kuulid on otsas, siis mängib pikk tühja relvaga tulistamise animatsioon, kus mängija võtab relva välja, sihib ja vajutab päästikule, kuid relv ei tulista. Kui tulistamissisendit vajutatakse enne animatsiooni lõppemist uuesti, siis mängib järgmisena lühike tulistamisanimatsioon, vastavalt laskemoonale kas laetud või tühja relvaga, kus mängija hoiab relva juba ees ja ainult vajutab päästikule. Võitluse sisemist loogikat tutvustatakse peatükis 5.3.3.

4.1.4 Surm ja taassünd

Mängija surma korral toimub taassünd, kus mängija viiakse viimasena külastatud kontrollpunkti ja mängu seis taastatakse viimase kontrollpunkti ajal olnud seisu. Kontrollpunktide loogikat tutvustatakse peatükis 5.2.6.

4.2 Loo süžee

Mängu peategelane on mees nimega Jack Moretti. Jacki õde Layla on kolm päeva teadmata kadunud olnud. Jack läheb Layla koju juhtlõngu otsima Layla kadumise kohta. Jacki saabudes Layla koju algab mängu tegevus.

Järgides Layla kodust leitud vihjeid jõuab Jack asustusest eraldatud metsa. Läbi mängu kestuse tuleb ilmsiks, et metsas pesitseb sekt, mille liikmed jumaldavad ebaloomulikku olendit, keda nad nimetavad "Äratajaks" (Awakener). Sektil paistab olevat kinnisidee metsas viibimisega ja nad ei ole nõus metsast lahkuma, ega ka lubama kellelgi teisel, kes on metsa jõudnud, sealt lahkuda.

Läbi kohtumiste sekti liikmete ja nende maha jäetud päeviku fragmentide tuleb välja, et ka Layla on "Ärataja" mõju all ja ka teda ei saa enam usaldada. Lõpus on mängija valiku ees, kas Layla tapab Jacki või Jack tapab Layla. Ei ole õnnelikku lõppu, on vaid valik kahe tragöödia vahel.

“Awakeni” lugu on inspireeritud õuduskirjanike Stephen King ja Howard Philips Lovecraft teostest. Lovecrafti “Cthulhu kutsest” [17] pärineb idee üleloomulikest olenditest, keda sektid ülistavad ja kellel on mõistmatu võim reaalsuse üle. Stephen Kingi “Pikas rohus” [18] on inspiratsioon asukohale, mis hoiab mingi kontrolliga inimesi lahkumast. Samuti pärineb “Pikast rohust” idee teadmistest, mis ajavad inimesed hulluks.

4.3 Alad

Selles alapeatükis kirjeldatakse mängu kolme põhilist asukohta: Layla kodu, mets ja kirik. Igaüks neist aladest jaotub väiksemateks jaotisteks, nagu erinevad ruumid, õuealad või ehitised.

4.3.1 Layla kodu

Layla kodu on mängu esimene ala. Seal on kuus tuba: garaaž, elutuba, teise korruse vahetuba, köök, vannituba ja magamistuba (vt Joonis 6).

Joonis 6. Layla magamistuba

Layla kodus on hulganisti interakteeruvaid objekte, millest osal on ka mängu loos tähtis koht. Külmkappi avades ja nähes, et see on toitu täis, mõtleb Jack endamisi, et Layla lahkus kodust kiiruga. Magamistoa seinal on märge, mille peal on kirjas Layla sülearvuti parool. Sülearvuti avamine on vajalik, et mängus edasi metsa liikuda

4.3.2 Mets

Mets on mängu suurim ala. Mets koosneb sujuvalt looklevast maastikust, puudest, kahest metsamajast (vt Joonis 7) ja kiriku fassaadist.

Joonis 7. Metsamaja fassaad

Metsas leiab aset kolm nägemust ning neli dialoogi. Metsamajadest leiab mängija päevikufragmente, mille abil mängu lugu välja tuleb. Metsas näeb mängija kahes kohas hetkeks kedagi teda jälgimas. Kohatava tegelase Anthony dialoog annab palju infot mängu loo kohta. Kaks vaenlast ründavad mängijat. Metsas toimub ka mängu viimane stseen, kus Jack kohtub Laylaga.

4.3.3 Kirik

Kolmas peamine mängu ala on kirik. Kirikus tuleb välja detaile mängu loost ja on paar šokeerivat stseeni. Kirik koosneb neljast ruumist: kirikusaal (vt Joonis 8), kaks ruumi pööningul ja kelder.

Joonis 8. Kirikusaal

Kirikus satub mängija kokku kahe vaenlasega. Pööningu esimeses toas on kapp, millest mängija leiab surnukeha. Pööningu teises toas leidub päevikufragment, mis selgitab natuke metsa sekti motivatsioone. Keldris, pärast kohtumist teise vaenlasega, saab mängija relva.

4.4 Kasutajaliides

Mängija valikute tegemiseks ja vajaliku info edasi andmiseks on mängul kasutajaliides. Kasutajaliidesesse kuuluvad mängu peamenüü (vt Joonis 9), pausimenüü, sätete aken, teated, juhised sisendi kasutamiseks, seljakotiliides (vt Joonis 5) ja kasutajasisendi liides.

Joonis 9. Peamenüü

Kasutajaliidese disainimisel pöörati tähelepanu stiililisele sobivusele mängu meeleolu ja kunstiga. Seetõttu kasutab kasutajaliides madala resolutsiooniga fonti. Kuna ülejäänud mängus kusagil hiirt ei kasutata, on ka kasutajaliides navigeeritav vaid läbi klaviatuuri või mängupuldi, hiire sisendeid kasutajaliideses ei ole.

5. Arendus

Selles peatükis kirjeldatakse mängu arendusprotsessi. Antakse ülevaade töövoost ja erinevate tööriistade kasutamisest. Kirjeldatakse alade disaini, mängusüsteemide loomist, varade hankimist, ise loomist ja kohendamist, varjutajate loomist ja kasutamist ning mehaanikate algoritme.

5.1 Alade disain

Alade disaini "Awakenis" võib jaotada neljaks suuremaks tööplokiks: maastik, objektid, taust ja valgustus. Iga ala disain hõlmas rohkemal või vähemal määral igaüht mainitud jaotistsest.

5.1.1 Maastik

Metsa õuekeskkonnas on kasutusel Godoti pistikprogramm (ingl *plugin*) SmartShape2D, mis võimaldab luua kahemõõtmelises keskkonnas sujuvalt looklevat maastikku (vt Joonis 10). SmartShape2D sõlm (ingl *node*) *SS2D_Shape* võtab parameetritena maastiku serva- ja täitetekstuuri, milleks metsa stseenis valiti vastavalt muru ja mulla tekstuurid. SmartShape2D võimaldab loodud *SS2D_Shape* sõlmele genereerida maastikku järgiva *CollisionPolygon2D* sõlme, mida kasutatakse kokkupõrgeteks (ingl *collision*) tegelastega ja füüsikale alluvate objektidega.

Joonis 10. SmartShape2D pistikprogrammiga loodud sujuv maastik

Mängu tubastes keskkondades on maastik horisontaalne ja sirge. See on saavutatud Godoti *TileMapLayer* sõlmega, mille abil saab *tileseti* jaotisi ruudustikus mängumaailma maalida. Iga sirge maastikuga ala stseenis on Godoti *StaticBody2D* tüüpi sõlm, mida kasutatakse nii põranda kui seinte kokkupõrgeteks mängu füüsikale alluvate objektidega ning tegelastega.

5.1.2 Objektid

Mängumaailma elavamaks muutmiseks lisati aladesse palju erinevaid objekte. Siseruumides on neiks erinevad mööbliesemed, nõud, raamatud ja elektroonilised seadmed. Metsa lisati loodusobjekte, nagu puud, põõsad, kivid ja lõke. Kindlate objektide täpseks positsioneerimiseks kasutati *Sprite2D* sõlmi. Kui oli vaja asetada palju objekte, mille joonised asusid samas *tileset* failis, oli otstarbekam kasutada *TileMapLayer* sõlmi.

Osa objektidest on mängus interakteeruvad. Enamjaolt kasutati sellisel juhul kaht veidi erinevat koopiat samast objektist, mis asuvad üksteise peal ja mida muudetakse vastavalt objekti seisule nähtavaks ja nähtamatuks. Interakteeruvaid objekte tutvustatakse lähemalt peatükis 5.2.5.

5.1.3 Taust

Alade stseenidel on kaht tüüpi taustasid: parallaksefektiga ja staatilised taustad. Parallaksefektiga taustadel on erinevate kiirustega kerivad kihid, et luua illusioon ruumilisusest. Staatilised taustad ei liigu mängumaailmas ja vastavalt stseeni suurusele kas on paigal või kerivad ekraanil muu mängumaailmaga samal kiirusel kaasa kaamera liikudes. Osades stseenides kombineeritakse staatilisi ja parallaksefektiga taustasid, näiteks staatiline siseruumi sein, kus aknast paistab parallaksefektiga õuetaust.

Parallaksefektiga taustad on loodud *ParallaxBackground* sõlmega. *ParallaxBackground* sõlm võtab kihtidena ühe või mitu *ParallaxLayer* sõlme, millest igaühele antakse tekstuur. Igal kihil saab siis määrata kerimiskiiruse osana ülejäänud mängumaailma kerimiskiirusest (kui kiirus on määratud kui 1, on tegemist efektiivselt staatilise taustaga). Väiksemad kiirused loovad illusiooni, et kiht asub kaugemal ja suuremad kiirused loovad illusiooni, et kiht asub lähemal. Nii saab kiirust suuremaks kui 1 määrates ning kihi mängumaailma peale tõstes luua esiplaani (vt Joonis 11).

Joonis 11. Parallaksefektiga taust (taga rohelised kihid) ja esiplaan (ees tumedad puud)

Staatilised taustad on loodud kas *Sprite2D* sõlmega, kui taust koosneb ühest suurest joonisest või *TileMapLayer* sõlmega, kui taust koosneb varieeruvast ruudustikust. *Sprite2D* sõlme kasutab näiteks kirikusaal, kus on korratud mitu korda sama seinasegmenti. *TileMapLayer* sõlme kasutab näiteks sein Layla kodus garaažis, kus telliskivide välimus on loodud ruudustiku maalimisega.

5.1.4 Valgustus

Mängus on kasutusel sujuv kahemõõtmeline valgustus. Selleks kasutati sõlme *PointLight2D*. Valgustus on lisatud enamikus alades mängijale, et hämaras keskkonnas oma lähedal paremini näha, valgusallikatele, nagu lambid ja aknad ja kindlatel juhtudel ka mujale mängija tähelepanu köitmiseks. Konkreetsete valgusvihkude loomiseks kasutati mõnedes olukordades ka sõlme *LightOccluder2D*, et luua varju efekt (vt Joonis 12).

Joonis 12. Valgustus mängija ümber ja valgusvihuga laelamp

Sujuv valgustus on võimas tasemedisaini (ingl *level design*) tööriist, millega on võimalik mängule anda meelepärast meeleolu, atmosfääri ja karakterit. Sujuval valgustusel oli "Awakeni" alade disainis tähtis roll.

5.2 Mängusüsteemid

Mängu loogika, protsessid ja igasugune käitumine on tagatud paljude erinevate üksteisest sõltuvate mängusüsteemidega. Selles alapeatükis antakse tehniline ülevaade "Awakeni" arenduses disainitud põhilistest mängusüsteemidest.

5.2.1 Signaalid

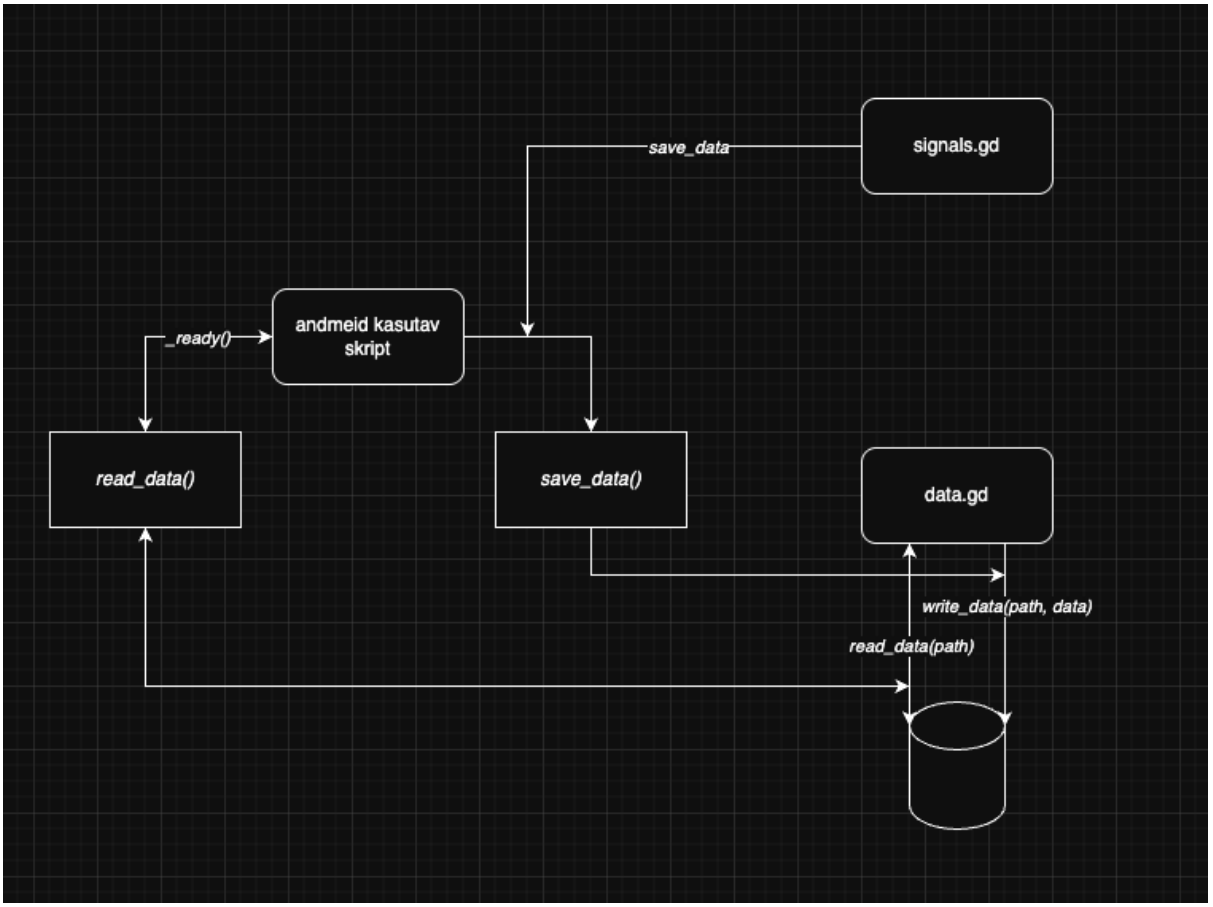
Godot kasutab ühe sõlmede vahelise suhtlemise mehhanismina signaale. Kindlaid signaale saadavad kindlad sõlmed automaatselt, näiteks *AnimationPlayer* saadab animatsiooni lõppemisel signaali *animation_finished(anim_name: StringName)*. Signaale on võimalik ka ise skriptides luua märksõnaga *signal* ja neid saata käsuga *emit_signal(signal: StringName)*. Skriptis on võimalik signaalile vastav reaktsioon defineerida käsuga *connect(signal: StringName, callable: Callable)*. Erinevad mängu skriptid saadavad kindlaid signaale, millele teised skriptid reageerivad, näiteks skript *interactable.gd* saadab signaali *interacted,* millele näiteks skript *car.gd* reageerib funktsiooniga *_on_interactable_interacted()*.

Ülemänguliste signaalide, mida võivad saata ja millele võivad reageerida erinevad skriptid, haldamiseks on *autoload* skript *signals.gd*. Kui skript on muudetud *autoloadiks*, siis hoitakse seda mälus kogu mängu jooksmise vältel ja sellele pääsevad ligi kõik mängu osad. Skripti *signals.gd* kasutatakse globaalselt info edastamiseks, näiteks kui on vaja mängu seisu salvestada või teavitada skripte, et mängija korjas seljakotti eseme.

5.2.2 Salvestussüsteem

Mängu seisu salvestamiseks ja taastamiseks on vajalik andmete andmekandjale salvestamine ja nende sealt lugemine. Mängu andmeid hoitakse andmekandjal JSON formaadis. Andmete salvestamise ja lugemise haldamiseks on *autoload* skript *data.gd*, mille funktsioone kutsuvad välja kõik skriptid (v.a *dialogue_controller.gd*), millel on vaja andmeid salvestada või lugeda.

Andmeid salvestatakse enne stseeni muutumist. Stseen muutub ehitistesse sisenedes ja neist väljudes, eri tubade vahel liikudes ja autoga Layla kodu ja metsa vahel liikudes. Enne stseeni muutmist käivitatakse käsk *Signals.emit_signal('save_data')* (*autoload* skriptid on Godotis suure algustähega). Igas asjakohases skriptis on defineeritud tee failisüsteemis vastava andmefailini ning *_ready()* funktsioonis on käsud *read_data()* ja *Signals.connect('save_data', save_data)* (*_ready()* on Godoti sisseehitatud funktsioon, mis käivitub sõlme mälusse laadimisel). Igas skriptis *read_data()* funktsioon kasutab *data.gd* skripti andmete lugemiseks ja *save_data()* funktsioon kasutab *data.gd* skripti andmete salvestamiseks (vt Joonis 13).



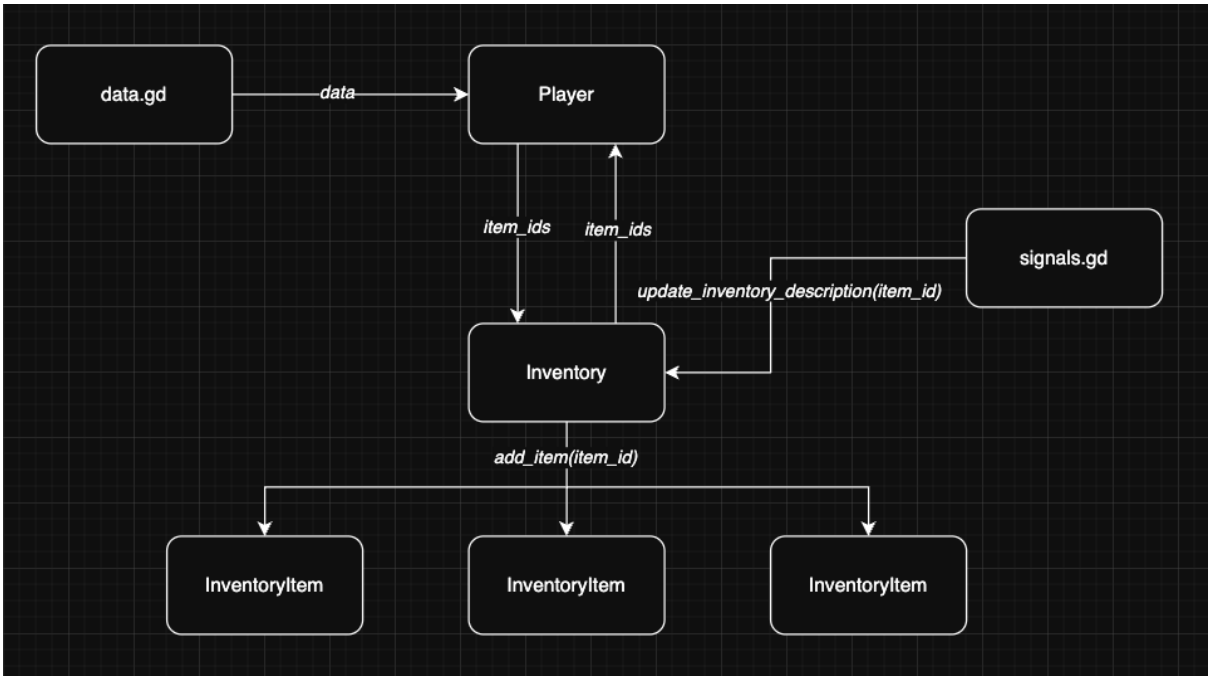
Joonis 13. Salvestussüsteemi töövoog

Skriptis *data.gd* on andmete lugemiseks funktsioon *read_data(path: String)*, mis loeb etteantud asukohast andmed ja tagastab need sõnastiku andmetüübina. Andmete andmekandjale kirjutamiseks on funktsioon *write_data(path: String, data: Dictionary)*, mis kirjutab etteantud asukohta sõnastiku andmetüübina saadud andmed. Andmete algväärtustele taastamiseks on funktsioon *reset_data()*, mida kasutatakse uue mängu loomisel. Kontrollpunkti jõudmisel käivitub funktsioon *create_checkpoint()* ja mängija surma korral käivitub funktsioon *reset_to_checkpoint()*. Surma ja kontrollpunktide loogikat kirjeldab põhjalikumalt peatükk 5.2.6.

5.2.3 Seljakott

Mängus leiduvate üleskorjatavate esemete haldamiseks on seljakotisüsteem. Seljakoti kasutajaliidest ja osa loogikast haldab stseen *Inventory*. Iga seljakotis leiduv ese on isend stseenist *InventoryItem*. Mängija stseen *Player* haldab nimekirja esemete ID-dest, mille põhjal luuakse *Inventory* stseenis *InventoryItem* isendid.

Skriptis *inventory.gd* käivitatakse *_ready()* funktsioonis funktsioon *read_item_map()*, mis loeb *data.gd* skripti abil andmekandjalt sõnastiku. Selle sõnastiku võtmeteks on esemete ID-d ja väärtusteks on omakorda sõnastikud, kus on defineeritud vastavalt vajadusele kolm või neli väärtust. Võtmele *title* vastav väärtus kuvatakse esemete loetelus vastava eseme tekstuuri kõrval (vt Joonis 5). Võtmele *description* vastav väärtus annab mängijale infot eseme kohta ja võib seeläbi kanda edasi ka mängu lugu. Võtmele *texture* vastab tee mängu failisüsteemis esemele seljakoti kasutajaliideses vastava tekstuurini. Mõnede esemete sõnastikes on ka võti *secondary_description*, mille väärtus on alternatiivne kirjeldus. Kui mängus leiab aset sündmus, mille peale peaks mingi eseme kirjeldus seljakotis muutuma, käivitatakse käsk *Signals.emit_signal('update_inventory_description', item_id)*. Skriptis *inventory.gd* käivitatakse *_ready()* funktsioonis käsk *Signals.connect('update_inventory_description', _on_update_inventory_description)*. Signaaliga ühendatud funktsioonis *_on_update_inventory_description(item_id)* kirjutatakse mälus hoitavas esemete sõnastikus vastava eseme *description* väärtus üle *secondary_description* väärtusega. Selline funktsionaalsus väljendab mängijakarakteri uue informatsiooni teada saamist. Seljakoti hetkeseisu hoitakse massiivi näol, mis koosneb esemete ID-dest. Funktsioon *add_item(item_id)* lisab etteantud ID esemete massiivi *item_ids* ja loob *InventoryItem* isendi lisatud ID-le vastavate väärtustega (vt Joonis 14).



Joonis 14. Seljakotisüsteemi töövoog

Skriptis *inventory_item.gd* on defineeritud neli muutujat: *texture*, *title*, *description* ja *id*. Skripti *inventory.gd* funktsioonis *add_item(item_id)* määratakse neile muutujatele väärtused. Muutuja *texture* väärtuse muutumisel määratakse *InventoryItem* stseeni *TextureRect* sõlmele vastav tekstuur. Muutuja *title* väärtuse muutumisel määratakse stseeni *Label* sõlmele vastav tekst. Seljakoti eseme valimisel küljel asuvast nimekirjast kuvatakse suurelt eseme detailvaade, kus võetakse pildi jaoks tekstuur *InventoryItem texture* muutujast ja kirjeldus *description* muutujast.

Skriptis *player.gd* kutsutakse *read_data()* funktsioonis pärast *data.gd* skriptist andmete saamist välja käsk *inventory.set_item_list(data['inventory'])*, kus muutuja *data* hoiab endas sõnastiku kujul mängijakarakteri andmeid. Andmetes vastab võtmele *inventory* massiiv hetkel seljakotis olevate esemete ID-dest. Skripti *inventory.gd* funktsioonis *set_item_list(item_ids)* tühjendatakse massiiv *self.item_ids*, kustutatakse hetkel eksisteerivad *InventoryItem* objektid, itereeritakse läbi etteantud massiivi *item_ids* ning kutsutakse iga elemendi puhul välja käsk *add_item(item_id)*. Olukordades, kus on vaja kontrollida, kas mingi ese leidub seljakotis, kutsutakse skriptis *player.gd* välja funktsioon *has_item(item_id)*. Selles funktsioonis kontrollitakse, kas skripti *inventory.gd* muutuja *item_ids* sisaldab hetkel etteantud ID-d. See on vajalik näiteks lukustatud ust avada üritades, et kontrollida vastava võtme olemasolu.

5.2.4 Dialoogisüsteem

Mängu kaasahaaravamaks muutmiseks, loo edastamiseks ja atmosfääri loomiseks disainiti dialoogisüsteem. Dialoogide visuaalset poolt haldab stseen *Dialogue* ja sisemist loogikat haldab stseen *DialogueController*. Stseen *Dialogue* on vaikimisi nähtamatu ja muudetakse nähtavaks dialoogi alustades. Dialoogides kuvatakse ekraanile kõneleja nimi ja tähthaaval räägitav tekst (vt Joonis 15).

Joonis 15. Dialoog mängija ja tegelase nimega Anthony vahel

Skriptis *dialogue_controller.gd* on *@export* märgisega muutuja *dialogue_node*, mis viitab kasutatavale *Dialogue* sõlmele stseenis. Muutujale *@export* märgise lisamine muudab muutuja dünaamiliselt väärtustatavaks läbi Godoti graafilise kasutajaliidese. Dialoogi ridade indeksit hoitakse muutujas *i*. Muutuja *dialogue_path* hoiab teed projekti failisüsteemis vastava dialoogi andmeteni. Muutuja *dialogue_data* hoiab massiivi kujul dialoogi andmeid. Funktsioon *set_dialogue_path(path)* väärtustab muutuja *dialogue_path* ja kutsub välja funktsiooni *read_dialogue_data()*. Funktsioonis *read_dialogue_data()* loetakse andmekandjalt *dialogue_path* määratud asukohast andmed ja salvestatakse muutujasse *dialogue_data*. See funktsioon ei kasuta skripti *data.gd*, sest erinevalt muudest loetavatest andmetest on dialoogide andmed salvestatud massiividena, mitte sõnastikena. Kuna massiivi näol loetakse andmeid ainult ühes kohas, ei olnud otsest vajadust skripti *data.gd* selleks funktsionaalsust luua ning iseseisvama (ingl *self-contained*) variandina kirjutati dialoogide andmekandjalt lugemise kood otse *dialogue_controller.gd* skripti. Funktsioon *begin_line()* loeb muutujast *dialogue_data* positsioonilt *i* väärtused *speaker*, *text* ja *delays*, edastab need *dialogue.gd* skriptile, suurendab *i* väärtust ühe võrra ning kutsub välja käsu *dialogue.begin_text()*. Funktsioon *start_dialogue()* annab muutujale *i* väärtuseks 0, muudab *Dialogue* sõlme nähtavaks ning kutsub välja käsu *begin_line()*. Funktsioon *interrupt_dialogue()* katkestab käimas oleva dialoogi poole pealt. Seda kasutatakse juhul, kui

mingi mängija tegevus või mängu sündmus vajab järsku dialoogi lõpetamist, näiteks kui mängija liigub kiriku keldris liiga lähedale teda relvaga sihtivale mehele ja mees tulistab.

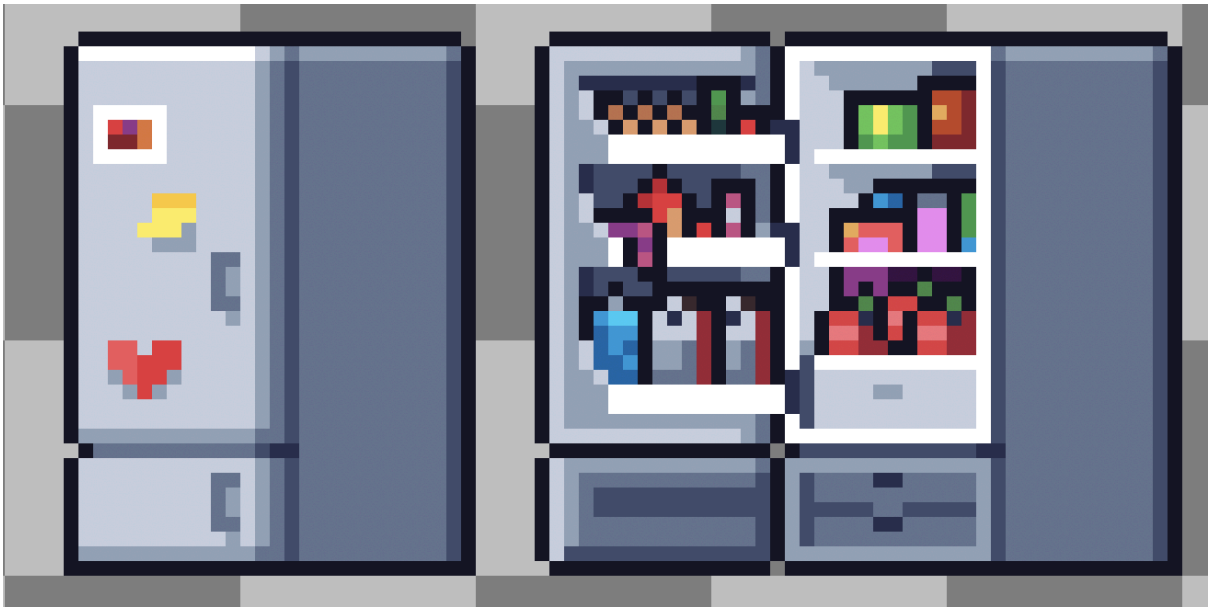
Skriptis *dialogue.gd* on defineeritud muutuja *DEFAULT_DELAY*, *@export* märgisega muutuja *advance_by_input* ja muutuja *i*. Funktsioonis *begin_text()* antakse muutujale *i* väärtuseks 0, kuvatakse ekraanile rääkija nimi *speaker* väärtusena ning kutsutakse välja käsk *add_text()*. Funktsioonis *add_text()* kontrollitakse, kas *i* väärtus leidub sõnastikus *delays* võtmena. Kui leidub, siis määratakse taimeri ajaks sõnastikust vastav väärtus, vastasel juhul määratakse ajaks *DEFAULT_DELAY* väärtus. Seejärel lisatakse kuvatavale dialoogile üks karakter *text* muutujast, suurendatakse muutujat *i* ühe võrra ja käivitatakse taimer. Taimeri lõppemisel kontrollitakse, kas *text* muutujas on veel kasutamata karaktereid ning vastavalt kas kutsutakse *add_text()* välja uuesti või saadetakse signaal *line_ended*. Varieeruvad viivitused *delays* sõnastikus on vajalikud, et edastada visuaalselt pause tegelase kõnes. Signaali *line_ended* korral kontrollitakse, kas *advance_by_input* on tõene. Kui *advance_by_input* on tõene, kuvatakse pärast väikest pausi ekraanile vihje edasi liikumiseks nupu vajutamiseks ja dialoog ei liigu edasi enne mängija sisendit. Vastasel juhul liigub dialoog järgmise rea juurde edasi kohe peale rea lõppemist. Ilma sisendita dialoogi kiiresti edasi liikumine on vajalik näiteks selleks, et edastada tegelase paanilist ja hirmunud kõnet.

5.2.5 Interakteeruvad objektid

Mängumaailma elavamaks muutmiseks on kasutusel palju interakteeruvaid objekte. Interakteeruvad objektid võivad omada tähtsust loo või mängus edasi liikumise jaoks, kuid võivad olla ka ilma sügavama eesmärgita osa interakteeruvast maailmast. Enamik interakteeruvad objektid kasutavad *ObjectVariationController* stseeni.

Stseen *ObjectVariationController* muudab mängija interaktsiooni korral interakteeruva objektiga objekti peal kuvatavat veidi erinevat variatsiooni nähtavaks ja nähtamatuks (vt Joonis 16). Skriptis *object_variation_controller.gd* on defineeritud *@export* märgisega muutujad *object_variation*, *thoughts*, *oneshot* ja *item_pickup*. Andmekandjalt loetakse väärtused muutujatele *status*, *invokes_thoughts* ja *active*. Muutuja *object_variation* hoiab viita objekti variatsiooni sõlmele stseenis, mida muudetakse nähtavaks ja nähtamatuks. Mängija interaktsiooni korral kutsutakse välja funktsioon *_on_player_interacted()*. Selles funktsioonis kontrollitakse, kas *active* on tõene, ning sel juhul muutetakse *status* väärtus vastupidiseks (*status = not status*), kutsutakse välja käsk *upate_status()*, kui

invokes_thoughts on tõene, muudetakse see vääraks ja kutsutakse välja sisemine monoloog massivi *thoughts* sisuga, kui *oneshot* on tõene, muudetakse *active* vääraks, ning kui *item_pickup* on tõene, saadetakse signaal *item_pickup(item_id)* seljakotti eseme korjamiseks. Funktsioonis *update_status()* muudetakse objekti variatsioon nähtavaks või nähtamatuks vastavalt *status* muutujale (*object_variation.visible = status*). Andmete salvestamisel salvestatakse muutujate *status*, *invokes_thoughts* ja *active* hetkeseisud andmekandjale.



Joonis 16. Külmkapi baas (vasakul) ja variatsioon (paremal)

Keerulisemad interakteeruvad objektid ei kasuta *ObjectVariationController* stseeni, sest neil on igaühel väga spetsiifiline käitumine ja nende loogika ühte skripti koondamine ei olnud optimaalne. Selliste objektide käitumine on programmeeritud nende enda skriptides. Selline käitumine hõlmab näiteks erinevate pikkustega animatsioone, mis käivituvad interaktsiooni korral, osakeste (ingl *particles*) käitumist ja muud loogikat. Keerukama loogikaga interakteeruv objekt on näiteks televiisor Layla kodu esimesel korrusel.

5.2.6 Kontrollpunktid

Mängija surma korral viiakse mängija tagasi viimasesse kontrollpunkti ja taastatakse viimast kontrollpunkti läbides olnud salvestatud andmete seis. Kontrollpunkte realiseerib stseen *Checkpoint*.

Stseen *Checkpoint* põhineb Godoti *Area2D* sõlmel, mis tuvastab mängija sisenemist kontrollpunkti ja käivitab skriptis *checkpoint.gd* seepeale funktsiooni

_on_body_entered(body). Funktsioonis *_on_body_entered(body)* kutsutakse välja käsk *Data.create_checkpoint()* ja käsk *queue_free()*, mis eemaldab sõlme stseenist. Skriptis *data.gd* funktsioon *create_checkpoint()* kopeerib mängu andmete kausta *data/* hetkesisu kausta *checkpoint/*.

Mängija surma korral käivitub käsk *Data.reset_to_checkpoint()*, milles kirjutatakse kausta *checkpoint/* sisu kausta *data/*. Siis laetakse mängu seis uuesti andmekandjalt ja mäng jätkub viimasena külastatud kontrollpunkti andmetega. Seeläbi on varasemale seisule taastatud seljakoti sisu, info selle kohta, millised tegelased on elus või surnud, mängija laskemoon ja muud mängu andmed.

5.2.7 Sisemine monoloog

Mängijakarakteri mõtete edastamiseks mängijale on "Awakenis" sisemise monoloogi süsteem (vt Joonis 17). Sisemine monoloog väljendab mängijakarakteri emotsionaalset seisu, vihjab järgmistele tegevustele mängus ja annab edasi lugu. Mõtteid kuvab ekraanile *Player* stseenis Godoti *Label* tüüpi sõlm *Thoughts*.

Joonis 17. Sisemine monoloog

Kui sündmus mängus või interaktsioon interakteeruva objektiga põhjustab mängijakarakteris mõtteid, kutsutakse välja käsk *Signals.emit_signal('big_brain_time', thoughts)*, milles

argument *thoughts* on sõnede massiiv, mis sisaldab mõtete järjestust. Skriptis *thoughts.gd* on *_ready()* funktsioonis käsk *Signals.connect('big_brain_time', _on_big_brain_time)*. Funktsioonis *_on_big_brain_time(thoughts)* mängitakse iga mõtte jaoks animatsiooni, kus mõtte tekst ilmub mängija pea kohale, püsib seal kaks sekundit ja siis hajub ära.

5.3 Mehaanikaalgoritmid

See alapeatükk annab tehnilise ülevaate algoritmidest, mis haldavad erinevaid mängu mehaanikaid. Kirjeldatakse olekumasinate kasutamist, liikumise loogikat ja võitlusmehaanika loogikat.

5.3.1 Olekumasinad

Üks levinud tehnika videomängudes tegelaste käitumise orkestreerimiseks vastavalt mängijasisendile või tehisintellektile on olekumasinad. Olekumasinaid kasutavad "Awakenis" mängijakarakter ja mõned vaenlased.

Olekumasinat kasutaval tegelasel on määratud võimalikud olekud, neile igaühele vastav käitumine ja hetkene olek. "Awakenis" on olekumasinad implementeeritud *enumidena*, mis määravad võimalikud olekud. Hetkest olekut hoitakse muutujas *state*. Funktsioonis *_physics_process(delta)* kontrollitakse hetkest olekut ja kutsutakse välja vastav funktsioon (vt Joonis 18). Godoti funktsiooni *_physics_process(delta)* kutsutakse välja kord iga mängukaadri puhul.

```
func _physics_process(delta: float) -> void:
    if not dead:
        match state:
            IDLE:
                idle_state()
            MOVE:
                move_state(delta)
            HIT:
                hit_state()
            SHOOT:
                shoot_state()
            CHECK_PHONE:
                check_phone_state()
            DEAD:
                dead_state()
            FLOAT:
                float_state()
```

Joonis 18. Olekumasin skriptis *player.gd*

Olek võib muutuda vastavalt sisendile, interaktsioonile mängumaailmaga või skriptitult. Näiteks skriptis *player.gd* on vaikimisi olek *IDLE* ning liikumissisendi puhul muutub olek väärtuseks *MOVE*. Olekumasinad tükeldavad erinevad käitumismustrid eraldi koodiplokkideks, muutes koodi loetavamaks, kergemini hooldatavaks ja skaleeritavaks.

5.3.2 Liikumine

Mängija sujuvaks liikumiseks on skriptis *player.gd* defineeritud muutujad *walk_speed*, *run_speed*, *accel*, *friction* ja *running*. Kui mängija on olekus *MOVE*, käivitub funktsioon *move_state(delta)*. Argument *delta* saadakse funktsioonist *_physics_process(delta)* ja see on ühe mängukaadri pikkus sekundites (ehk kaadrisageduse pöördväärtus). Kasutades argumenti *delta* saab tagada ühtlase liikumiskiiruse, kiirenduse ja aeglustuse erinevate kaadrisageduste korral.

Funktsioonis *move_state(delta)* kuulatakse sprintimissisendit (klaviatuuril vaikimisi “Shift”, puldil “L2”), mille peale muudetakse muutuja *running* väärtus vastupidiseks (*running* = *not running*) ning määratakse liikumiskiirus kas *walk_speed* või *run_speed* väärtuseks vastavalt muutuja *running* väärtusele. Kui mängija hoiab all liikumissisendit ja hetkekiirus on alla määratud liikumiskiiruse, kiirendatakse mängijakarakterit vastavalt *accel* väärtusele kuni liikumiskiiruse saavutamiseni ning siis hoitakse ühtlast kiirust. Kui suund muutub,

kiirendatakse liikumiskiirust vastassuunas kuni liikumiskiiruse vastandväärtuse saavutamiseni ning siis hoitakse ühtlast kiirust. Kui mängija laseb liikumissisendi lahti, aeglustatakse mängijakarakterit vastavalt *friction* väärtusele ning peatumisel antakse olekule *IDLE* väärtus. Kui hetkekiirus on suurem kui *walk_speed* väärtus, siis mängib jooksmisanimatsioon, vastasel juhul mängib kõndimisanimatsioon.

5.3.3 Võitlus

Vaenlaste elimineerimiseks on kaks võitlusmehaanikat: löömine ja tulistamine. Löömisvõime on mängijal mängu algusest peale. Tulistamise võimalus tekib mängu lõpu poole, kui seljakotti on lisandunud revolver.

Kui mängija kasutab löömissisendit (klaviatuuril vaikimisi "K", puldil "X"), läheb skript *player.gd* olekusse *HIT*. Seepeale käivitub funktsioon *hit_state()*, milles peatatakse horisontaalne liikumine ja mängitakse löömisanimatsiooni. Löömisanimatsioonis lülitatakse sisse tabamisala (ingl *hitbox*), mis tuvastab kokkupõrkeid ainult määratud sõlmedega, sealhulgas vaenlastega. Kui tabamisala tuvastab kokkupõrke sõlmega, saadetakse skripti *signals* kaudu signaal *camera_shake*, mille peale kaamera raputab lühidalt, ning käivitatakse käsk *body.get_hit()*, kus *body* on viit tuvastatud sõlmele. Iga löödava sõlme skriptis on defineeritud funktsioon *get_hit()*, mis võib teha erinevaid asju, aga iga olekumasinaga vaenlase puhul minnakse selle peale *DEAD* olekusse. Lüüa saab lisaks vaenlastele mõningaid objekte, millel on funktsioonis *get_hit()* vajadusele omane loogika. Näiteks metsa alguses peab mängija lööma väravat, et see avaneks ja saaks edasi liikuda.

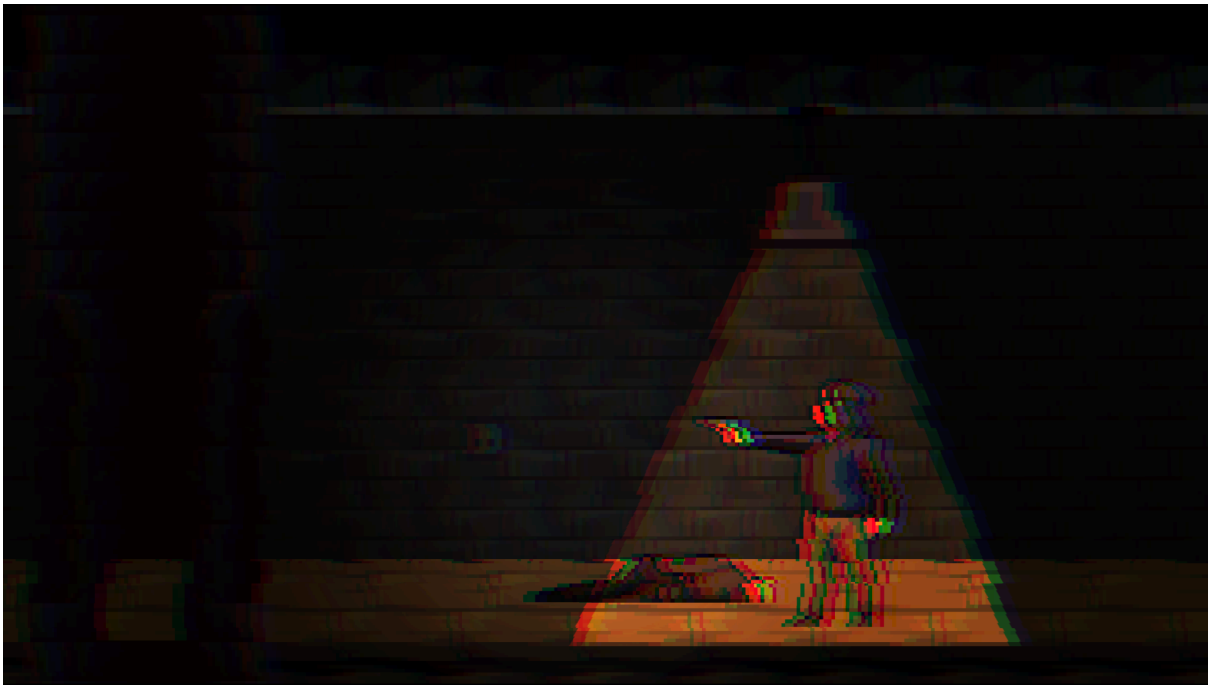
Olekule *SHOOT* on defineeritud neli sõne tüüpi muutujat: *shoot_first_anim*, *shoot_again_anim*, *dry_fire_first_anim* ja *dry_fire_again_anim*. Kui mängija vajutab tulistamissisendit (klaviatuuril vaikimisi "L", puldil "Y"), siis kontrollitakse, kas seljakotis on revolver ja kui on, liigub olekumasin *SHOOT* olekusse, mispeale käivitub funktsioon *shoot_state()*. Funktsioonis peatatakse horisontaalne liikumine ja mängitakse animatsiooni, mille nimi on salvestatud muutujasse *shoot_anim*. Muutuja *shoot_anim* väärtus on üks ülalmainitud muutujate väärtustest, *_ready()* funktsioonis antakse sellele väärtuseks kas *shoot_first_anim* või *dry_fire_first_anim* vastavalt sellele, kas relval on laskemoona. Animatsioonide *shoot_first_anim* ja *shoot_again_anim* käigus käivitatakse funktsioon *emit_bullet()*, mis vähendab laskemoona ühe võrra, lisab stseeni kuuli sõlme ja määrab vastavalt mängija suunale selle liikumissuuna. Kuuli sõlm liigub määratud suunas edasi kuni

ta kas põrkub kokku vaenlasega, mispeale kutsub välja käsu *body.get_shot()*, kus *body* on viit vaenlase sõlmele ja käivitab käsu *queue_free()*, või väljub kaamera piiridest, mispeale ta kutsub välja ainult käsu *queue_free()*. Kui mistahes *SHOOT* oleku animatsiooni jooksul vajutab mängija uuesti tulistamissisendit, antakse skriptis muutujale *shot_again* tõene väärtus. Mistahes *SHOOT* oleku animatsiooni lõppedes kontrollitakse kõigepealt, kas *shot_again* on tõene. Kui on, siis kontrollitakse, kas veel on laskemoona alles. Kui laskemoona veel on, väärtustatakse *shoot_anim* väärtusega *shoot_again_anim_anim*, vastasel juhul väärtusega *dry_fire_again*, ning funktsioon *shoot_state()* käivitub uuesti. Animatsioonide *shoot_again_anim* ja *dry_fire_again_anim* mõte on lühendada järjestikuseid animatsioone, nii et kohe uuesti tulistades ei tõmba mängijakarakter relva uuesti vöölt, vaid juba hoiab relva ees ja vajutab päästikule. Kui *shot_again* on väär, saab uueks olekuks *IDLE*.

5.4 Varjutajad

Visuaalsete efektide loomiseks on “Awakenis” kasutusel varjutajad (ingl *shader*). Newcastle Ülikooli kursuses defineeritakse varjutaja järgmiselt: “Varjutajad on lühikesed programmid, mis jooksevad otse graafikariistvaral, ja mis viivad läbi operatsioonid, mis muudavad graafilised andmed soovitud lõplikuks pildiks” [19:2]. Graafikariistvara suurel skaalal töö paralleelsust toetava arhitektuuri tõttu on varjutajad arvutuslikult odav viis pildi käitumist mõjutada.

“Awakeni” varjutajad on kirjutatud Godoti varjutamiskeeles (ingl *shading language*), mille faililaiend on *.gdshader*. Varjutajate abil on saavutatud lillakas mustvalge ekraan metsas kiriku ees aset leidvas nägemuses, tegelaste ja objektide ühevärviliseks muutmine, näiteks tume figuur metsa alguses kiiresti mööda jooksmas, piirjoon Layla ümber kiriku ees nägemuses ning pildi virvendamine surma ja viimase *cutscene*’i ajal (vt Joonis 19).



Joonis 19. Pildi virvendamise varjutaja mängija surma korral

Mitmes varjutajas on kasutusel *uniform* märgisega muutujad, mis muudavad varjutajate muutujad dünaamiliselt väärtustatavaks läbi graafilise kasutajaliidese. Efektiivselt on *uniform* varjutajate ekvivalentne märgis GDScript *@export* märgisega. Arenduse käigus kirjutati veel mõned varjutajad, mis ei jõudnud mängu lõplikku versiooni.

5.5 Mänguvarad

Selles alapeatükis kirjeldatakse mänguvarade (ingl *assets*) loomist, hankimist ja kohendamist. Kirjeldatakse eraldi kunstivarade, heliefektide ning muusika käsitamist.

5.5.1 Kunstivarad

Enamik “Awakenis” kasutatud kunstivarad on internetist kas tasuta või tasu eest hangitud CC litsentsiga, mis lubab varade kasutamist nii isiklikel kui ärilistel eesmärkidel [20]. Enamike varade autorid ei nõua viitamist, kuid austusest autorite ja nende töö vastu on mängu lõputiitrites kõik autorid mainitud.

Kuigi enamik kunstivarad on alla laetud, oli paljudel neist vaja teha muudatusi. Muudatused hõlmasid näiteks spetsiifiliste animatsioonide juurde joonistamist, objektide eemaldamist joonistelt, objektide lisamist, paranduste tegemist ja muud. Osa kunstivaradest, sealhulgas

kõik kasutajaliideses kasutatud varad, loodi algusest lõpuni ise. Kunstivarasid töödeldi *pixel art* töötlustarkvaras Aseprite.

Ühe kunstivarade töötlemise meetodina kasutati skaleerimist. Skaleerimine selle töö kontekstis on pildi resolutsiooni muutmine, et saavutada soovitud pikslitihedus. Skaleerimise abil on võimalik näiteks muudes stiilides joonistused *pixel art* stiili teisendada. Skaleerimine ei ole täiuslik ja jätab pildile visuaalseid vigu, näiteks läbipaistvad või muul moel kohatud pikslid. Läbipaistvate pikslite eemaldamise automatiseerimiseks kirjutati Pythoni skript *remove_transparent.py*, mis võtab sisendiks *.png* pildi ja eemaldab sealt kõik pikslid, mille alfaväärtus on alla määratud piirväärtuse. Piksli alfaväärtus väljendab piksli läbipaistvust, kus väärtus 0 vastab täiesti läbipaistvale ja 255 vastab täiesti läbipaistmatule pikslile. Alfaväärtuse piirväärtus on skriptis vaikimis 255, seega eemaldatakse kõik pikslid, mis ei ole täiesti läbipaistmatud. Pärast läbipaistvate pikslite skaleeritud pildist eemaldamist tehti veel täpsemad parandused käsitsi Aseprite tarkvaras.

5.5.2 Heliefektid ja muusika

Kõik "Awakenis" kasutatud heliefektid on internetist tasuta alla laetud CC litsentiga. Kõik heliefektid pärinevad Pixabay veebilehelt, millele viidatakse mängu lõputiitrites. Mängu muusika on autori enda kirjutatud ja kitarril mängides salvestatud.

Muusikat ja heliefekte töödeldi helitöötlustarkvaras Logic Pro. Heliefektide töötlemine hõlmas pikematest helifailidest soovipäraste osade välja lõikamist, müra eemaldamist, kõrvade ärritamise vältimiseks järskude helide alguste ja lõppude sujuvaks muutmist ning helide algusest ja lõpust vaikuse eemaldamist. Muusika töötlemine hõlmas samuti müra eemaldamist ja alguste ja lõppude sujuvaks muutmist, aga ka heli kompresseerimist ja muusika eri osadest kokku lõikamist.

6. Testimine

Mänguarenduse lahutamatu osa on testimine. Testimise käigus lastakse projektiga mitteseotud isikutel mängu mängida ja anda tagasisidet. Nii saavad arendajad teha mängus muudatusi, et mäng oleks sihtgrupile kaasahaaravam ja mängul oleks suurem võimalus äriliselt õnnestuda. Selles peatükis antakse ülevaade "Awakeni" testimise metoodikatest, tulemustest ja testimistulemustest orienteeritud mängu viidud muudatustest.

6.1 Metoodikad

Laialt võib "Awakeni" testimise jaotada kahte tasemesse: alfatestimine ja beetatestimine. Kursuse "Tarkvara testimine" [21] järgi on alfatestimine testimine, mida viivad läbi tarkvara loonud organisatsiooni liikmed, kes ei ole ise osalenud projekti arenduses. "Awakeni" puhul osalesid alfatestimises autori sõbrad ja pereliikmed ning lõputöö juhendaja. Kursuse "Tarkvara testimine" järgi [21] on beetatestimine testimine, milles osalevad tarkvara lõppkasutajad. "Awakeni" beetatestimine toimus avalikult läbi interneti ning selles osalesid kõik soovijad.

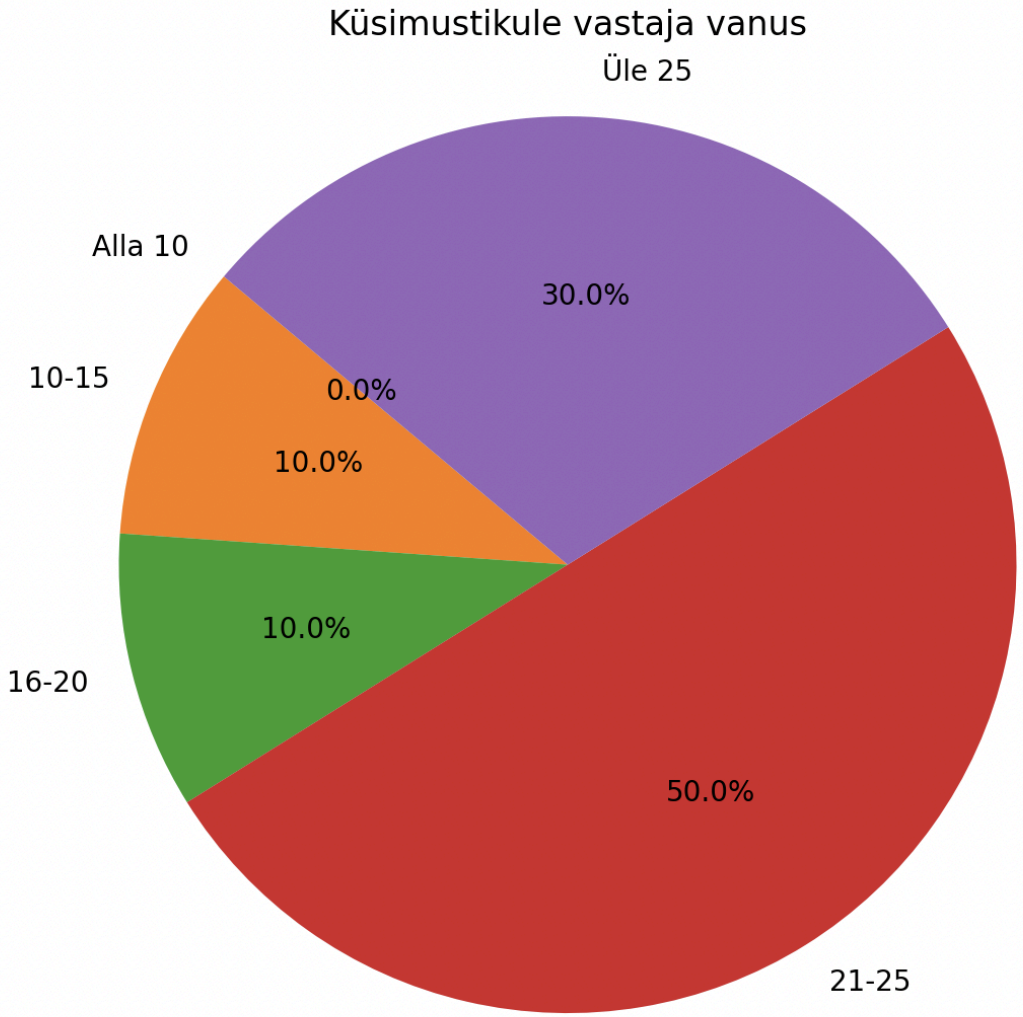
Alfatestimine toimus üsna mitteformaalselt. Sõpradega aega veetes või pere külastades lasti huvilistel "Awakeni" hetkeseisu otse arendusmasinas mängida ja katsetada. Mängijate suulise tagasiside põhjal viidi siis mängu sisse muudatused. Alfatestimine oli ka efektiivne viis vigade avastamiseks ning nende parandamiseks enne mängu avalikustamist.

Beetatestimine oli formaalsema metoodikaga ja viidi läbi interneti teel. Mäng avalikustati *early access* olekus tasuta veebilehel itch.io. Mängu avalikustamislehele lisati link küsitlusele (vt Lisa II). Küsitlusele vastamine oli vabatahtlik ja anonüümne. Osadel küsimustel oli vastus skaala vormis ühest viieni. Sellised küsimused uurisid mängijatelt, kui kaasahaarav "Awaken" neile oli, kui sujuv oli kogemus mehaaniliselt, kui nauditav oli mäng visuaalselt, kui õudne "Awaken" oli ja kui kaasahaarav oli mängu lugu. Ülejäänud küsimused olid vabas vormis tekstiliste vastustega. Nende küsimustega sai soovi korral selgitada oma skaala kujul vastuseid, teavitada avastatud vigadest, väljendada aspekte, mis erakordselt kas meeldisid või ei meeldinud "Awakeni" juures, soovitada, kas mängu eest on mõtet raha küsida, ning väljendada midagi muud, mida vastaja lisada soovis.

6.2 Tulemused

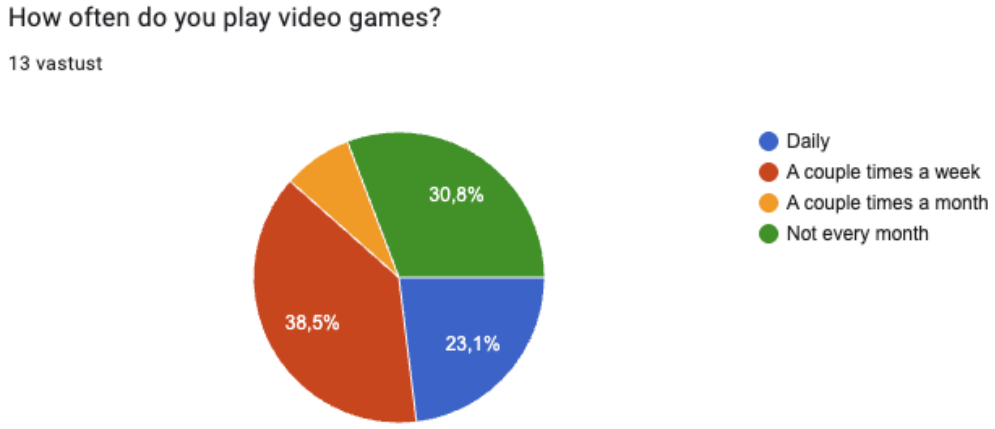
Alfatestimise käigus tuli ilmsiks hulganisti vigu. Esines visuaalseid, kogemust võrdlemisi vähe häirivaid vigu, kuid ka mängu jätkamist takistavaid vigu (ingl *game breaking bug*). Lisaks vigadele andsid testijat tagasisidet, mis kohad olid nende arvates liiga segadusseajavad, igavad või keerulised. Samuti anti tagasisidet heliefektide, valgustuse ning kasutajasisendi kohta. Peale suulise tagasiside sai ka testijaid mängides vaadates infot selle kohta, kui mõni osa mängust võib osutuda liiga keeruliseks või segadusseajavaks, näiteks kui mängija on pikka aega kinni mingis kindlas kohas. Samas peab alfateste analüüsides meeles pidama, et osa alfatestijaist ei langenud ilmselt "Awakeni" sihtgruppi, sest oli testijaid, kes ei ole harjunud videomänge mängima, näiteks arendaja vanemad ja tüdruksõber. Sellistele testijatele võivad teatud kohad osutuda keerulisemaks kui regulaarsetele mänguritele. Enim said alfatestimise käigus kriitikat ebaselged juhised mängus ja kasutajasisendi kaardistamisotsused. Kiitust anti visuaalidele, interakteeruvusele ja heliefektidele.

Beetatestimise küsimustikule vastas 13 inimest. Neist neli olid naissoost ja üheksa olid meessoost. Vastajad jaotati vanusegruppidesse alla 10, 10-15, 16-20, 21-25 ja üle 25. Enamik vastajad osutusid vanematesse vanusegruppidesse (vt Joonis 20).



Joonis 20. Beetatestijate vanuse jaotus

Videomängude mängimise sagedus jaotus vastajate vahel üsna mitmekesiselt (vt Joonis 21). See on tulemuste analüüsiks kasulik, sest mitmekesine valim annab hea ülevaate erinevate kasutajatüüpide eelistustest. Muid 2D külgvaates õudusmänge peale "Awakeni" oli 13 vastajast varem mänginud kaks. Väike proportsioon on üsna oodatav, sest 2D külgvaates õudusmängud on üsna spetsiifiline žanr ja enamik inimesed ilmselt sellega kokku puutunud ei ole.



Joonis 21. Beetatestijate videomängude mängimise harjumused

Beetatestimises ilmnes üksikuid vigu. Tagasiside küsimistiku vastustes toodi vigade raporteerimise kohas esile probleeme, mis ei osutunud otseselt vigadeks, vaid rohkem isiklikeks eelistusteks (vt Lisa II). Sellegi poolest on mängijais frustratsiooni tekitavad mängu osad probleemid, millega peaks tegelema. Kasutaja Spudcats Youtube'i videos ilmnes tõsiseid visuaalseid vigu [22]. Nende parandamine osutus väga keeruliseks, sest teadaolevalt on Spudcats ainus testija, kellel sellised vead esinevad ja autoril ei ole õnnestunud probleeme taastekitada.

"Awakeni" üleüldist kaasahaaravust hindasid testijad pigem kõrgeks. Skaalal ühest viieni oli keskmine vastus 4,46. Mehaaniline reageerivus sai samuti enamjaolt positiivset tagasisidet. Skaalal ühest viieni oli keskmine vastus 4,38. Visuaale hinnati skaalal ühest viieni keskmise vastusega 4,77. Viletsamaks hinnati "Awakeni" õudust. Skaalal ühest viieni oli keskmine vastus 3,31. Loo kaasahaaravust hinnati samas üsna kõrgelt. Ühest viieni skaalal oli keskmine vastus 4,62.

Beetatestimise käigus sai enim kriitikat mängu tempo. Metsas ei leidnud mitme vastaja jaoks aset piisavalt palju sündmusi. Kriitikat said ka kasutajasisendi kaardistused. Enim kiitust said visuaalid ja lugu. Vastajate arust sobis piksliline kunstistiil mängu sisuga ja lugu oli põnev.

Küsimustiku lõpus uuriti ka beetatestijatelt, kui palju, kui üldse, oleksid nad nõus "Awakeni" lõpliku versiooni eest maksma. Vastused jäid vahemikku 5-20 eurot. Tasub märkida, et

mõned vastajad võisid hinda märkides eeldada, et lõplik mäng tuleb märgatavalt suurema skoobiga kui *early access* versioon.

6.3 Muudatused mängus

Vastavalt testimise tagasisidele viidi mängu sisse mõned muudatused. Kasutajaliidese kaardistamise kriitikale vastuseks on "Awakeni" lõplikus versioonis kasutajasisend mängija poolt konfigureeritav. Ajakavas püsimise huvides ei lisatud metsa juurde sündmusi. Testijaid jälgides aga tuli ilmsiks, et üks sündmus metsas jäi paljudele testijatele märkamata. Mainitud sündmuses mängivad metsa kindlas kohas arusaamatud sosinad. Mängijate tähelepanu köitmiseks muudeti sosinad valjemaks.

Testimisfaasis mängust jäi ajapiirangute tõttu välja ka mängu muusika. "Awakeni" lõplikku versiooni lisati muusika. Autori lootuse järgi võivad sündmusteta ajavahemikud mängus tunduda põnevamad, kui taustal mängib köitev muusika.

7. Kokkuvõte

Lõputöö käigus disainiti ja arendati 2D külgvaates õudusmäng pealkirjaga “Awaken”. Õuduse žanr on videomängudes levinud, kuid enamasti luuakse tänapäeval õudusmänge 3D formaadis. 2D külgvaates formaadi kasutamine muutis “Awakeni” mõnevõrra omapärasemaks.

Inspiratsiooni saamiseks ja unikaalsuse tagamiseks analüüsiti olemasolevaid 2D külgvaates õudusmänge. Lähemaks käsitlemiseks valiti “Lone Survivor”, “The Coma” seeria mängud ja “Limbo”. Sellised valikud tehti nende mängude edukuse ja sarnasuste tõttu “Awakeniga”.

Lõputöös kirjeldati valminud mängu põhimehaanikaid, lugu, mängu alasid ja kasutajaliidest. Anti tehniline ülevaade mängu loomisprotsessist, sealhulgas alade disainist, põhilistest loodud mängusüsteemidest, algoritmidest mehaanikate taga, varjutajate kasutamisest ja mänguvarade hankimisest, loomisest ja töötlemisest.

Arenduse vältel lasti kõrvalistel isikutel mängu testida. Tagasiside “Awakenile” oli enamjaolt positiivne. Kiideti visuaalset köitvust ja mängu lugu. Kriitikat anti mängus esinevatele vigadele ja mängu tempole. Aeg-ajalt leidsid testijad, et mängus on tegevustest ja sündmustest puudus. Vastavalt tagasisidele tehti palju veaparandusi ja muid muudatusi mängus. Tulevikus on võimalik tempo parendamiseks veel mängu sisu lisada. Lõputöö ajapiirangute tõttu jäi see parendus praegu tegemata.

Töö eesmärk oli luua kaasahaarav 2D külgvaates õudusmäng. Arvestades mängu valmimist ja enamjaolt positiivset tagasisidet loeb autor eesmärgi täidetuks.

8. Tänusõnad

Autor tänab lõputöö valmimisele kaasa aidanud isikuid. Töö juhendaja Madis Vasser aitas tagasisidega, soovitustega ja ligipääsuga salvestustehnikale. Kõik testijad panustasid tagasisidega mängu paremaks muutmisesse. Liisa Alavere panustas palju “Awakeni” turunduskampaaniasse sotsiaalmeedias.

Viidatud kirjandus

[1] Williams, S.Computer Horror-Game Demographics: 57 Stats. 2023. Vaadatud 09.05.2025 https://techpenny.com/horror-game-demographics/

[2] Epps, D.A. From Splatterhouse to Resident Evil, horror games found the fun in fear. *Gaming*. 2021. Vaadatud 09.05.2025 https://www.digitaltrends.com/gaming/horror-game-evolution/

[3] Vikipeedia. Alone in the Dark (1992 video game). Vaadatud 05.05.2025 https://en.wikipedia.org/wiki/Alone_in_the_Dark_(1992_video_game)

[4] "Lone Survivor: The Director's" Cut müügileht Steam koduleheküljel (2025). Vaadatud 22.03.2025 https://store.steampowered.com/app/209830/Lone_Survivor_The_Directors_Cut/

[5] "The Coma" seeria müügileht Steam koduleheküljel (2025). Vaadatud 22.03.2025 https://store.steampowered.com/bundle/45978/The_Coma_Triple_Threat/

[6] "The Coma 2: Vicious Sisters" müügileht Steam koduleheküljel (2025). Vaadatud 22.03.2025 https://store.steampowered.com/app/1045720/The_Coma_2_Vicious_Sisters/

[7] "Limbo" müügileht Steam koduleheküljel (2025). Vaadatud 22.03.2025 https://store.steampowered.com/app/48000/LIMBO/

[8] Godot mängumootori kodulehekülg (2024). Vaadatud 07.12.2024 https://godotengine.org/

[9] Mohd, T.K., Bravo-Garcia, F., Love, L., Gujadhur, M. ja Nyadu, J. (2023). Analyzing Strengths and Weaknesses of Modern Game Engines. *International Journal of Computer Theory and Engineering*, *15*(1), 54-60. doi:10.7763/IJCTE.2023.V15.1330

[10] Holfeld, J. (2024). On the relevance of the Godot Engine in the indie game development industry

[11] Aseprite kodulehekülg (2024). Vaadatud 08.12.2024 https://www.aseprite.org/

[12] Aseprite müügileht Steam koduleheküljel (2024). Vaadatud 08.12.2024 https://store.steampowered.com/app/431730/Aseprite/

[13] Logic Pro müügileht Apple App Store koduleheküljel (2024). Vaadatud 08.12.2024 https://apps.apple.com/gb/app/logic-pro/id634148309

[14] Visual Studio Code repositoorium GitHubi koduleheküljel (2024). Vaadatud 08.12.2024 https://github.com/microsoft/vscode

[15] Khleel, N.A.A. ja Nehéz, K. (2020). Comparison of Version Control System Tools. *Multidiszciplináris tudományok*, *10*(3), 61-69. doi:10.35925/j.multi.2020.3.7

[16] Final Cut Pro Apple App Store koduleheküljel (2025). Vaadatud 05.05.2025 https://apps.apple.com/us/app/final-cut-pro/id424389933?mt=12

[17] Lovecraft H. P. Cthulhu kutse. Eesti: Fantaasia. 2017.

[18] King S., Hill J. In the Tall Grass. Ameerika Ühendriigid: Esquire. 2012.

[19] Newcastle Ülikool. Introduction to Graphics Programming (2018). Vaadatud 15.04.2025 https://research.ncl.ac.uk/game/mastersdegree/graphicsforgames/introductiontographics/IntroductionToGraphics.pdf

[20] Creative Commons kodulehekülg (2025). Vaadatud 15.04.2025 https://creativecommons.org/share-your-work/cclicenses/

[21] Tartu Ülikool. Tarkvara testimine (2025). Vaadatud 25.04.2025 https://courses.cs.ut.ee/2025/SWT2025/spring/Main/Loengud?action=download&upname=SWT2025_lecture06v03.pdf

[22] Spudcats. Awaken - It's You're Inner Jack (2025). Vaadatud 25.04.2025 https://www.youtube.com/watch?v=KkwtEBlEObU

Lisad

I. Terminid

1. **mängumootor** - tarkvararakendus, mida kasutatakse videomängude arendamiseks
2. **vabavaraline (tarkvara)** - avalikult leitava lähtekoodiga
3. **versioonikontroll** - süsteem failide jagamiseks ja nende ajaloo talletamiseks
4. **parallaks** - visuaalne efekt, kus lähemal asuvad objektid paistavad liikudes kiiremini mööduvat kui kaugemal asuvad objektid
5. **sõlm (Godot terminoloogias)** - mingit kindlat funktsiooni täitev objekt mängus; stseenipuu fundamentaalne komponent
6. **stseen (Godot terminoloogias)** - organiseeritud kogum sõlmedest, mis moodustavad mängus ühe loogilise terviku, näiteks tegelase, taseme või kasutajaliidese osa
7. ***cutscene*** - skriptitud filmiline sündmus mängus, kus kasutajasisend on piiratud või eemaldatud, ja mille eesmärk on tavaliselt edastada mängu lugu
8. **mängujämm** - mänguarenduse üritus, mille osalejad loovad lühikese aja jooksul (näiteks üks nädal või nädalavahetus) väikesi videomänge, tavaliselt võistlusformaadis; ingl *game jam*

II. Kaasnevad failid

Kaasnev failiarhiiv sisaldab järgmisi jaotisi:

1. **testimine** - testimisküsimustik ja selle vastused
2. **mäng** - käivitatav mäng Windows ja macOS operatsioonisüsteemidele

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Uku Tonsiver

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **2D külgvaates õudusmäng "Awaken"**, mille juhendaja on Madis Vasser, reprodutseerimiseks eesmärgiga seda seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile;

4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Uku Tonsiver
10.05.2025