Aleksi Skantz, Mika Hassinen, Tapio Törrönen, Teemu Vaittinen, Jami Ristimäki

Inachus-innovaatioprojektin yhteenvetoraportti

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ohjelmistotekniikka

12.5.2016

Metropolia

Sisällys

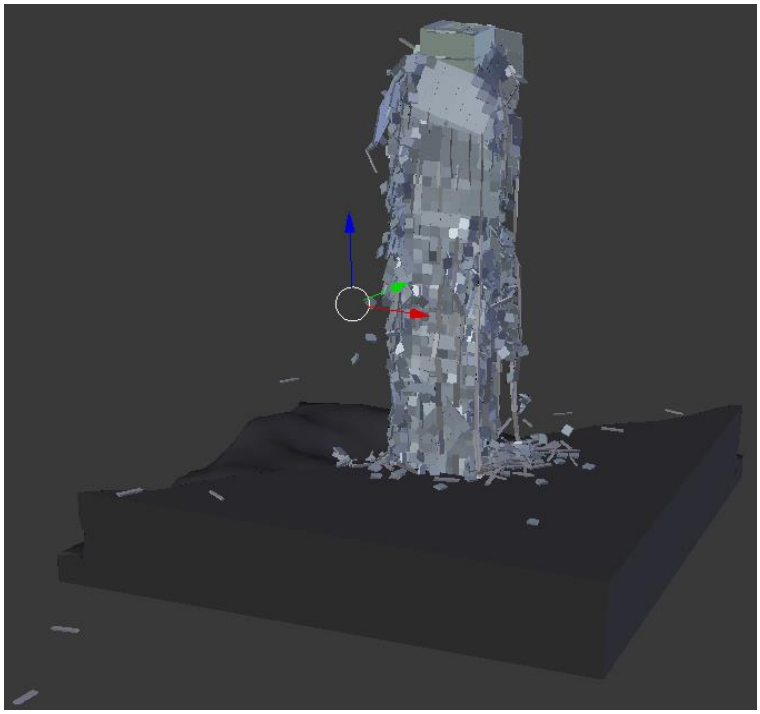
1 Johdanto

Projektin tarkoituksena oli kehittää asiakkaan kanssa sovellus joka liittyy heidän EU:n rahoittamaan Inachus-projektiin.

Inachus-projektissa kiinnostuimme heidän tekemistään 3D-simulaatioista. 3D-Simulaatiot olivat romahtavista kerrostaloista ja simulaatioiden idea oli tutkia m.m. minne ihmiset jäisivät jumiin ja mistä heidät voisi pelastaa jonkun katastrofin jälkeen.

Asiakkaalta saimme Blender-simulaatio ja mallinnus tiedostot jota rupesimme työstämään ja tutkimaan.
Blenderin käyttö osoittautui erittäin haastavaksi varsinkin kun kenelläkään ei ollut ohjelmasta kokemusta.

Monen mutkan jälkeen keksimme yhdistää aikaisemmat tiedostot Unreal Enginen. Päätimme siis viedä mallit pelimoottoriin ja tutkia mitä mahdollisuuksia tällä olisi meille tarjota.



Kuva 1. Alkuperäinen Blender-malli

2 Toiminnallinen ja tekninen -määrittely

2.1 Toiminnallinen määrittely

Lopputulokseksi halutaan saada aikaiseksi menetelmä jolla voidaan tuoda Blender-tiedostoja raunioista Unreal Engineen ja luoda siten mahdollisuus tutkia sitä peliympäristössä simulaationa. Tähän kuuluu mahdollisuus lisätä uhreja raunioihin joita etsiä ja antaa niille selviytymisarvoja tarpeen tullen. Lopuksi syntyy ohjelma jonka voi käynnistää tietokoneelta ilman erikoisohjelmia ja käydä läpi rauniot ihmishahmolla ja käärmerobotilla. Prosessi voidaan toistaa muille rauniomalleille.

2.2 Tekninen määrittely

Ohjelman on tarkoitus olla työkalu mallinnettujen raunioiden virtuaaliseen läpikäymiseen ja tutkimiseen. Ohjelman käyttämiseen tarvitsee tietokoneen ja muutaman ohjelman jotka peli yrittää asentaa mikäli ne puuttuvat. Lisäohjelmien asentaminen vaatii tosin järjestelmänvalvojan oikeudet. Ohjelman luomisessa käytetään Unreal Engine -ohjelmaa johon viedään Blender-mallinnusohjelmalla luotuja 3d-malleja romahtaneista rakennuksista. Unreal Engine on Epic Gamesin luontityökalu jota he jakavat ilmaiseksi rojalteja vastaan kun tekee tarpeeksi voittoa. Tämän jälkeen luodaan aloituspiste, ympäristö ja mahdolliset pelaajahahmot ja uhrit. Sitten käytetään Unreal Enginen export-ominaisuutta ja luodaan executable-ohjelma jonka käynnistää ilman että tarvitsee olla Blenderiä tai Unreal Enginea. Reunaehtoina mainittakoon että Unreal Engine voi olla vaikea käyttää jos koneella ei ole tarpeeksi suoritustehoa ja pahimmassa tapauksessa se ei toimi lainkaan.

3 Tuotteen esittely

Tuote on Unreal Enginellä tehty pelimäinen ohjelma, jossa voidaan tutkia Blenderissä luodun romahdus-simulaation lopputulosta joko lentävällä hahmolla tai “käärmerobotilla” joka pystyy ryömimään raunioiden koloissa ja löytämään uhreja.

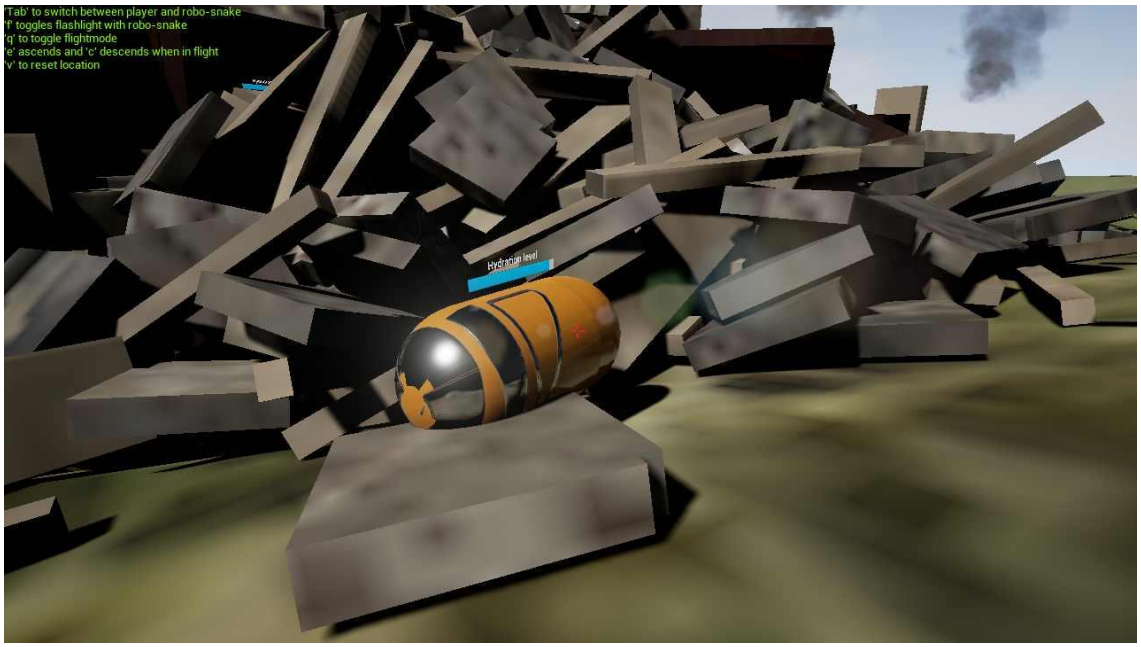
Pelin kenttänä toimii Blenderillä luodun romahdusmallin viimeinen vaihe jossa rakennus on romahtanut loppuun asti.

Peli alkaa hahmolla jonka on tarkoitus liikkua ja olla suurinpiirtein oikean ihmisen kaltainen. Hahmolla on kuitenkin ekstra ominaisuus joka antaa sen lentää tehden mallin ja kentän tutkimisesta helpompaa.

Käärmerobotti on toinen hahmo jota pääsee ohjaamaan nappia painamalla. Käärmerobotilla on valo jolla se osoittaa eteenpäin nähdäkseen pimeässä. Robotti on pieni että se voisi mahtua raunioiden pieniin rakoihin ja pystyy kiipeämään matalia tasoja edetäkseen. Käärmerobotti näkee aluksi vain ensimmäisestä persoonasta eteenpäin, mutta sillä on myös ominaisuus nähdä ympäristönsä kolmannesta persoonasta. Robotti pystyy myös pelastaa uhreja raunioista menemällä uhrin lähelle.

Kuva 2. Käärmerobotti

Raunioihin on ripoteltu uhreja kuvaamaan ihmisiä jotka ovat jääneet romahtaneen rakennuksen sisälle tai raunioiden lähelle. Uhrit pysyvät paikoillaan ja niille on annettu ominaisuuksia jotka kuvaavat niiden selviytymistä, kuten veden tasosta.



Kuva 3. Pelastettava uhri

4 Käyttöohje

Projektissa tuotetun ohjelman käynnistäminen ei edellytä Unreal Enginen asennusta. Kaikki projektitiedostot löytyvät https://github.com/tapix/Phoroneus -sivulta. Simulaatiopeli käynnistyy suoraan executable-tiedoston suorittamalla. Valmiin ohjelman executable -version käynnistämiseen tulee koneella olla asennettuna vähintään Direct X 11 -versio sekä Windowsin 2013 c++ redistributable package (niiden asennus vaatii järjestelmänvalvojan oikeudet).

4.1 Unreal Enginen asennusohjeet

Jotta pääset käyttämään UnrealEngine-kehitysympäristöä, laadi itsellesi kirjautumistunnukset ja rekisteröidy Epic Gamesin jäseneksi https://www.unrealengine.com/ -sivulla. Latauslinkin pitäisi löytyä vasemmasta yläkulmasta Epic Games -logoa klikkaamalla ja valitsemalla ‘Get Epic Games’.

4.2 Simulaatiopelin ohjeet

Peli alkaa Main menusta. Siinä voi valita käytettävän resoluution Options-valikon kautta tai kokonäytön tilan. Kun päävalikosta valitsee “Start”, alkaa varsinainen peli. Päävalikkoon pääsee pelissä Pause-menun kautta. Pause-menuun pääsee painamalla ‘m’ tai ‘Esc’ -näppäintä. Peli päättyy kun valitaan “Quit” kummasta tahansa menusta.

Pelaajaa ja robottikäärmettä ohjataan wasd-ohjaimilla, eli w:llä kävelee eteen päin, s:llä taakse päin, d:llä oikealle, a:lla vasemmalle. Hiiren liikkeillä käännetään katseen suuntaa. Tabulaattorilla vaihdetaan ohjataanko ihmispelaajaa vaiko robottikäärmettä, oletuksena ohjataan ihmispelaajaa pelin alkaessa. Kun robottikäärmettä ohjataan, näppäimellä ‘f’ laitetaan robottikäärmeen etuvalo päälle ja pois, oletuksena valo on

päällä. Robottikäärmettä ohjatessa näppäimellä ‘g’ vaihdellaan eri robottikäärmeen kuvakulmien välillä, ensimmäisellä painalluksella vaihdetaan käärmeen etukamerasta takakameraan, toisella kerralla vaihdetaan takakamerasta kolmannen persoonan kameraan ja kolmas painallus vaihtaa kolmannen persoonan kamerasta taas etukameraan.

Ihmispelaajaa ohjatessa näppäin ‘q’ laittaa lentoasetuksen päälle ja pois. Lentoasetus poistaa painovoiman vaikutuksen ihmispelaajalta, silloin ylöspäin liikutaan ‘e’-näppäimellä ja alaspäin liikutaan ‘c’-näppäimellä. Robottikäärmeellä ei ole lentoasetusta. Ihmispelaajan näkökulmaa voi vaihtaa ‘1’-näppäimellä ensimmäisen persoonan kamerasta kolmannen persoonan kameraan ja edestakaisin.

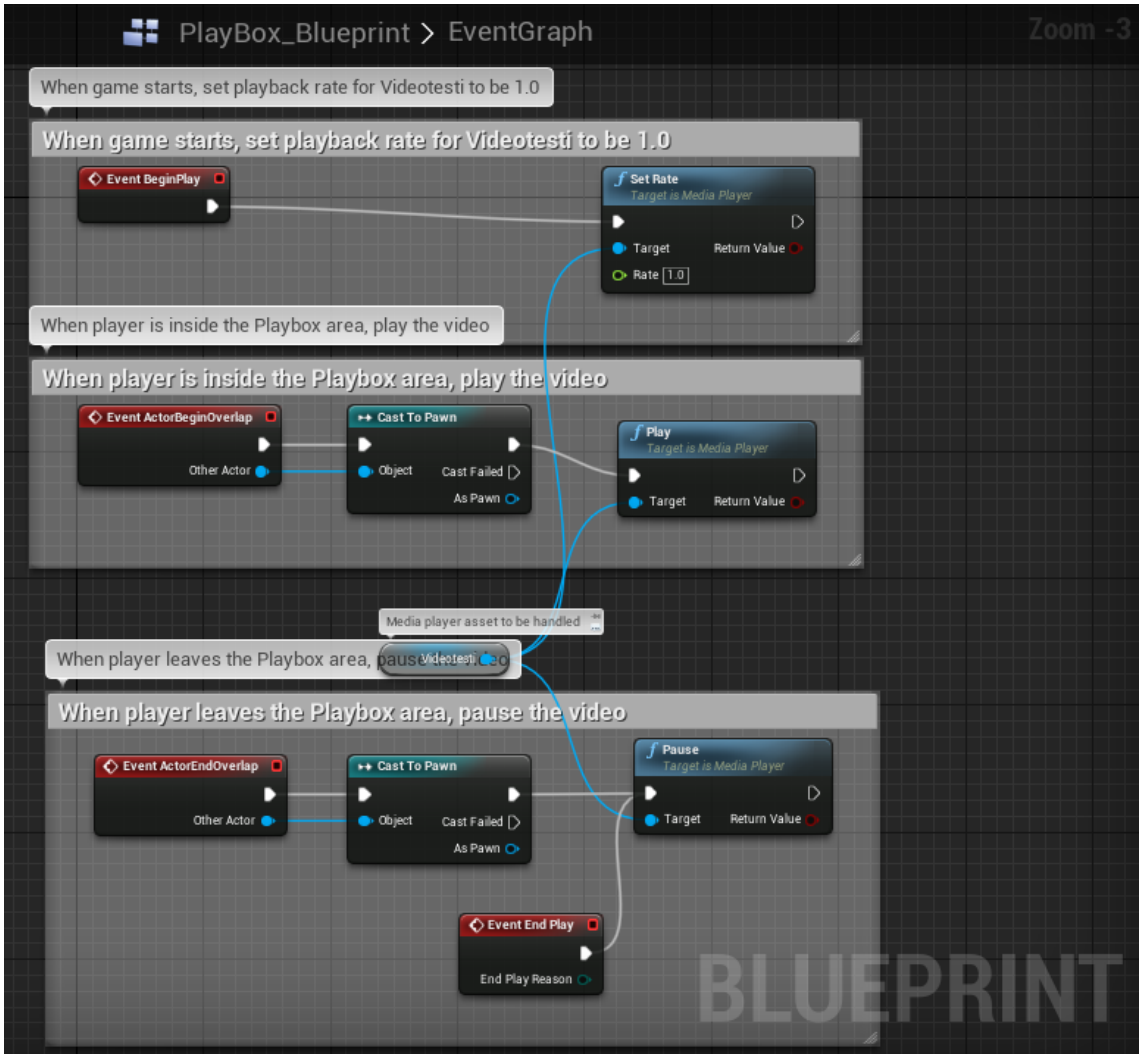
Ihmispelaajaa, robottikäärmettä, rakennusta tai maata sen ympärillä ei pelaaja pysty vahingoittaa. Näppäimellä ‘l’ laukaistaan räjähdys rakennuksen katolla, joka tiputtaa kuutioiden palasia ympäri kenttää. Jos robottikäärme tai ihmispelaaja jää jumiin johonkin rakennuksen osaan tai halutaan asettaa ihmispelaaja tai robottikäärme takaisin aloitusalustalle, onnistuu se ‘v’-näppäimellä, joka asettaa kontrolloidun hahmon takaisin alkuasentoon leijuvalle alustalle.

Rakennuksen ympäristöön on piilotettu neljä kapselin muotoista simuloidun onnettomuuden uhria, joiden nesteytystaso hupenee noin kahdeksassa minuutissa loppuun, sitä ennen uhrit pystyy “pelastaa” ohjaamalla joko robottikäärmeen tai ihmispelaajan uhrin viereen, kosketusetäisyydelle. Pelastettu uhri katoaa ja ilmoitus siitä tulee näytölle hetkeksi. Nesteytystason huvetessa nollaan uhria ei voi enää pelastaa.

4.3 Tekniset blueprint ohjeet

Pelilogiikka on sidottu Unreal Enginen Blueprint-tiedostoihin, joista keskeisimpiä ovat Simulation_body, Snake_blueprint, PlayBox_Blueprint, ThirdPersonCharacter, FirstPersonExampleMapin level blueprint sekä Main (menun) level blueprint. Suurin osa keskeisistä blueprinteistä on FirstPersonExampleMapin sisällä käytössä.

FirstPersonExampleMapin level blueprint sisältää logiikan pelattavan hahmon vaihtoon, pelaajan sijainnin resetoinnin sekä rakennuksen katolla tapahtuvan räjähdyksen logiikan. Mainin level blueprint sisältää main menun käynnistyslogiikan, mutta MainMenu ja PauseMenu -widgetit sisältävät suuren osan itse toiminnasta (Graph-puolella). ThirdPersonCharacter sisältää ihmishahmon liikkumisen ja kameran ohjaamisen logiikan sekä osaa Pause-menuun sidotusta pelin pysäyttämislogiikasta. Snake_blueprintissä on samaan tyyliin liikkumisen logiikka ilman lentoasetusta, eli sekin sisältää osan Pause-menuun sidotusta logiikasta. Simulation_body käsittää simulaatioruumiiden tilan logiikan ja käyttää Hydration_level -widgettiä näyttääkseen nesteytystason pelin aikana. Play_box on alue, minkä sisällä pelaajan pitää sijaita, että aloitusalustalla sijaitseva demovideo pyörii.



Kuva 4. Esimerkkikuva Blueprints –kaaviosta

5 Toteutumattomat ideat ja jatkosuunnittelu

Projektin aikana tuli useita mahdollisia kehitysideoita jotka eivät ajan puutteen vuoksi vain mahtuneet lopulliseen tuotteeseen.

Käärmerobotille oli suunniteltu infrapunakamera tai valonvahvistin. Infrapunakamera näkisi lämmönlähteitä ja uhreja esittäville kapseleille annettaisiin lämpöarvoja, jotka voisivat huveta kun uhrin elintasot putoavat pikkuhiljaa. Valonvahvistin toimisi kuten oikeassa elämässä ja antaisi mahdollisuuden nähdä paremmin raunioiden pimeissä loukuissa. Robotille suunniteltiin myös parempaa hitboxia. Robotin hitbox ei toiminut luontivaiheessa kuten haluttu ja jouduttiiin tyytymään pieneen epärealistiseen hitboxiin. Hitbox olisi voitu tehdä alusta lähtien laatikoksi jotta olisi saatu oikeanlaisempi osuman rekisteröinti.

Simulaatio uhreille oli suunniteltu lisäominaisuuksia kuten avunhuudot jotka kuuluisivat realistisesti huonommin mitä kauempana uhrista ollaan. Uhrien huudot jostain syystä kuuluivat joka paikassa missä vain joten ajan puutteessa huudot vain hiljennettiin kun ei tiedetty mistä ongelma johtuu ja kauanko sen korjaaminen kestää. Ei myöskään tiedetty onko asiakas edes kiinnostunut avunhuudoista.

Uhreille oli asetettu vain nesteytystaso joka vähenee pelin edetessä. Kun nesteytysmittari tyhjenee, uhri “kuolee” mutta ainoa kuoleman merkki on mittarin tyhjyys ja uhrilta katoaa törmäyksen tunnistus, tehden uhrin pelastamisesta mahdotonta. Suunnitteilla oli muitakin terveyden merkkien lisäyksiä, kuten happimittari, mahdolliset ruumiilliset vammat ja niiden aiheuttama mahdollinen verenvuoto tai valvetila. Eräs mahdollinen lisäys olisi myös kapselien vaihtaminen oikeiksi ihmismalleiksi jotka voisivat liikkua ja mahdollisesti olla jumissa romun alla joka pitäisi siirtää ennen kuin pelastus onnistuu. Esimerkiksi

jos uhri olisi koomassa päähän osuneen romun tai verenvuodon takia, ei tämä liikkuisi tai pystyisi huutamaan.

Uhrin kuolema-tilan voisi laukaista verenvuoto, lämmön puute, nestevajaus tai pään tuhoutuminen tai irtoaminen tai jos jokin terävä romu olisi läpäissyt tai leikannut irti tärkeitä elimiä kuten sydämen ja aivot. Kuoleman merkeiksi oli kapseleissa suunniteltu kapselin värin muuttaminen harmaaksi tai rastin laittaminen sen päälle. Oikeita ihmisiä kuvaavissa malleissa voisi käyttää samaa tai voisi pakottaa lähelle tulemisen ja tarkistamisen ennen kuin peli kertoo ihmisen tilanteen.

Robotille oli mietitty muita pelastusta auttavia ominaisuuksia kuten vesileilin käyttäminen uhrien nesteytystason auttamiseksi. Robotille suunniteltiin pyörien asentamista mikä tekisi sen liikkeistä ja käytöksestä hieman realistisempaa.

Peliin mietetittiin mahdollisuutta luoda uhrien paikat ja aloitus status sattumanvaraisesti. Mietittiin mahdollisuutta luoda joka aliohjelma tai ominaisuus joka laskisi robotille sopivia reittejä uhrien luo. Google Mapsista importattavat kartat ja niiden luominen peliin oli eräs idea. Google Mapsista voi mahdollisesti saada 3d-malleja rakennuksista myös. Useiden raunioiden importtaaminen suoraan Blenderistä Unreal Engineen oli ehdotus.

Eräs idea oli luoda peli serverinä niin että peliä voisi monipelata useampi henkilö samaan aikaan, kuten oikeassakin elämässä paikalla olisi paljon pelastushenkilökuntaa.

Raunioiden importtauksessa voisi olla jokaisesta rauniosta kolme mallia: alkuperäinen ehjä rakennus, romahtanut rakennus ja video josta näkee rakennuksen romahtamisen.

6 Yhteenveto

Unrealin ominaisuudet tarjosivat hyvät puitteet ja toteutusmahdollisuudet, joita pystyimme käyttämään hyväksi.

Projekti päättyi onnistuneesti ja ylitimme kaikki asiakkaan ja meidän itse asettamat tavoitteet.

Olimme myös tyytyväisiä siihen että saimme projektin hyvälle mallille ja sitä on helppo jatkaa kehittämään eteenpäin.