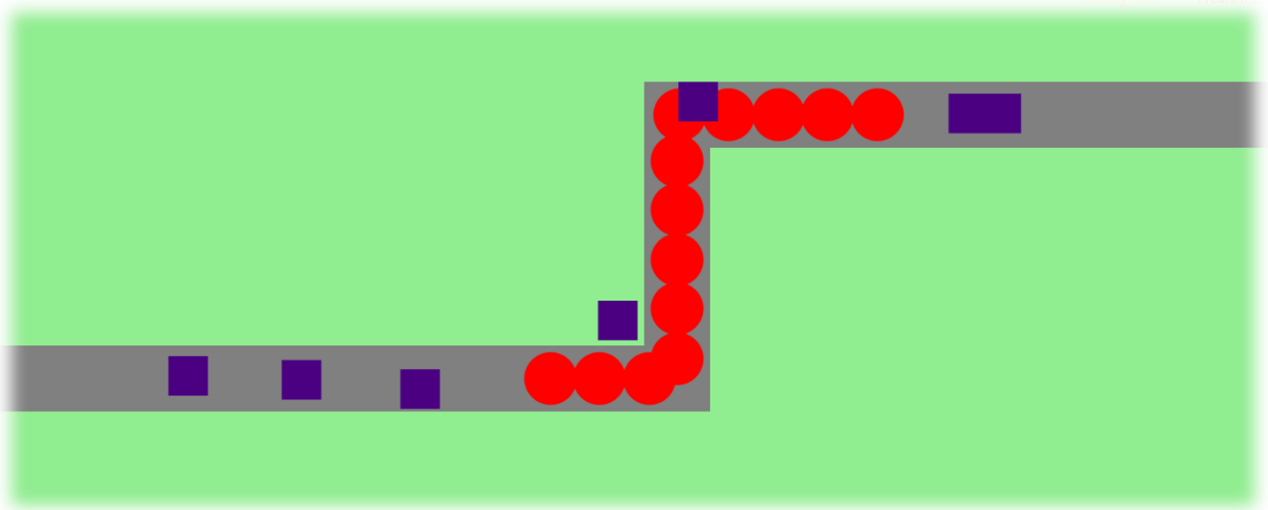
OS2 – Projektityön dokumentti



Anton Pirhonen, 653318, tietotekniikka kandidaatti, 2018, 28.4.2019

2. Yleiskuvaus

Tein Ohjelmointistudio 2 -kurssin projektina tornipuolustuksen. Suunnitelmani oli luoda peli, jossa vihollisia kulkee pitkin reittiä. Pelin tavoitteena on estää vihollisia saavuttamasta reitin loppua. Mikäli vihollinen kulkee reitin loppuun, niin pelaaja menettää osan elämäpisteistään. Peli on hävitty, jos elämäpisteet loppuvat. Peli koostuu tasoista, jotka koostuvat useammista aalloista.

Vihollisia on erilaisia: osa etenee nopeammin, osalla on enemmän elämäpisteitä. Kun tornit tai esteet vahingoittavat vihollista, niin sen elämäpisteet vähenevät. Kun vihollisen elämäpisteet loppuvat, niin se kuolee. Tällöin pelaaja saa rahaa, jolla voi ostaa uusia torneja.

Suuria muutoksia alkuperäisestä suunnitelmasta ei esiinny toteutuksessa. Poikkeamia ovat mm. tornien kehittämisen ja ammusten erikoivaikutusten puuttuminen. Tasoltaan projektini oli keskivaikea-vaativa. Se täyttää sille asetetut vaatimukset. Konfiguroitavuutta olisi tosin voitu selvästi lisätä mahdollistamalla myös reitin muokkaaminen tiedostosta. Tämänhetkisessä versiossa on mahdollista muokata vihollisaaltoja ja -tasoja sekä pelaajan elämäpisteitä ja rahoja.

3. käyttöohje

Ohjelma käynnistetään suorittamalla yksittäisolio GUI, joka on JFXApp. Tämän jälkeen pelaaja voi lisätä peliin torneja yläreunassa näkyvästä Towers-valikosta. Game menu-valikosta pelaaja voi ladata uuden pelin valitsemalla Load game- näppäimen kautta oikeassa formaatissa olevan tekstitiedoston. Game Menu:n Restart-valinta aloitta pelaajan sillä hetkellä pelaaman tason alusta. Peli päättyy jos viholliset tai elämäpisteet loppuvat. Tällöin pelaaja ohjataan voitto-/häviöruudulle, josta hän voi painamalla Play Again!-nappia palata uudelleen aloitetulle peliruudulle. Pelaaja voi luoda omia tasojaan projektin levels-kansioon. Tiedostojen on oltava oikeassa formaatissa (esitelty kohdassa 8).

4. Ohjelman rakenne

World on pelin maailmaa mallintava yksittäisolio. Se pitää kirjaa kaikissa pelimaailmassa esiintyvistä olioista, ja käsittelee niiden toimintoja ja vuorovaikutusta keskenään. Graafinen käyttöliittymä piirretään World:n pohjalta.

Tower on pelin tornia kuvaava luokka. Erilaiset tornityypit ovat tämän luokan aliluokkia. Torni osaa etsiä omalla alueellaan olevan vihollisen ja lähettää sitä kohti ammuksen, jota mallinnetaan Projectile-luokalla. Jokaisella tornityypillä on myös oma kumppaniolionsa, jota hyödynnetään vakioiden sijoituspaikkana sekä pattern matchingin kanssa.

Projectile kuvaa tornin lähettämää vihollista vahingoittavaa ammusta. Luokan ilmentymä etenee yksinkertaisesti suoraan kulkusuuntaansa ja vahingoittaa vihollista, jos sellaiseen osuu.

Abstrakti luokka Monster kuvaa pelissä esiintyvää vihollista. Sen aliluokat ovat erilaisia vihollistyyppejä. Jokaisella Monsterilla on metodit takeDamage ja advance, jonka avulla maailma päivittää vihollisen tilaa maailmassa. Kuten Tower-luokillakin, myös Monster luokilla on omat kumppaniolionsa joita hyödynnetään vakioiden säilyttämiseen ja pattern matchingiin. Monster etenee maailmassa reittiä kuvaavan Path-olion avulla.

Reittiä kuvaavalla Path-yksittäisoliolla on muuttuja path, joka koostuu pelin reitin muodostavista pisteistä. Reittiä ei voi muuttaa ilman ohjelmakoodin muokkaamista. Yksittäisistä pisteistä koostuvaa reittiä parempi valinta olisi ollut hyödyntää Grid-luokkaa vihollisten liikkumiseen. Path-yksittäisolion tiedostosta löytyy myös Grid-yksittäisolio, jonka tiles-muuttujassa on Tile-luokan ilmentymiä. Monstereiden reitti Path olisi järkevämpi toteuttaa Gridiin perustuen, mutta en ehtinyt muita luokkia Gridin kirjoittamisen jälkeen enää refaktoroimaan. Tällä hetkellä Gridin pohjalta piirretään kentän tausta. Tämä helpottaa mahdollisuutta teksturoida pelin ruutuja myöhemmin.

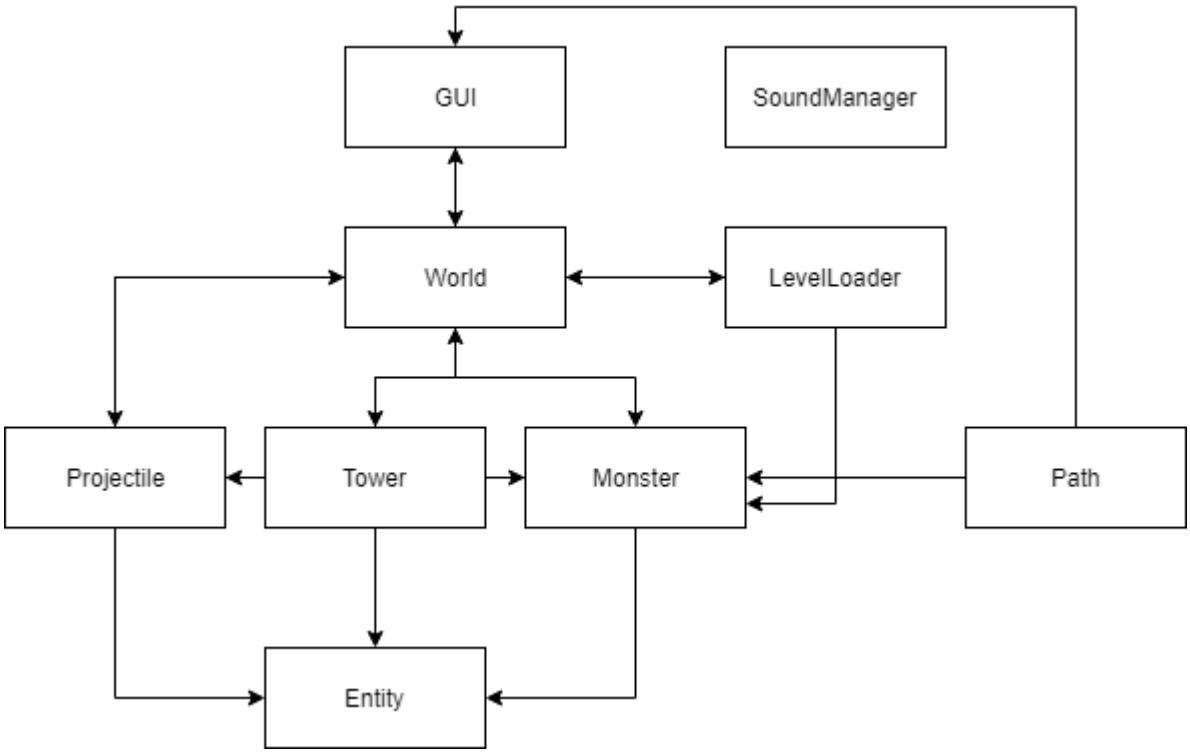
LevelLoader-yksittäisolio lataa pelin viholliset ja muut tiedot tiedostosta. Tekstiä käsittelemällä se muodostaa World:lle tasoja kuvastavia Wave-olioita.

Entity piirreluokka on maailmassa esiintyvien olioiden yhteinen piirreluokka. Sitä hyödynnetään piirtämisessä.

Pelin graafinen käyttöliittymä ja animointi tapahtuu GUI-yksittäisolion avulla. Keskeisin toiminta perustuu sovelluksen muuttujalle stage, jolle piirretään Scene. Sceneen lisätään keskeiset komponentit, jota ovat mm. yläkulman valikot. GUIssa on useita tapahtumankuuntelijoita, joiden avulla käyttäjän syötteestä lisätään olioita Worldiin sitä kautta ruudulle. GUI animoi Worldin sisällön AnimationTimer-luokassa, joka koneen kellon mukaan päivittää maailmaa. Lisäksi graafinen käyttöliittymä hyödyntää ScalaFX-kirjaston Alert-luokkaa, jonka avulla se ilmoittaa käyttäjälle virheellisestä tiedostosyötteestä.

SoundManager-yksittäisolio sisältää AudioClip-olioita, joiden play()-metodia kutsutaan äänten lisäämiseksi peliin.

Luokkarakenne on mielestäni toimiva. Pelimaailmassa olevia oliota kuvaavat luokat (Monster, Projectile, Tower) ovat toimivia, sillä niiden käyttäminen ja ilmentymien luominen on helppoa. Kokonaisuus on onnistunut myös Worldin, GUIn ja SoundManagerin kanssa. Jokainen yksittäisolio kuvastaa selkeää kokonaisuutta, ja niiden yhteistoimintaa on helppo hallita. Suurin kehityskohde on Path-yksittäisolio. Path tulisi refaktoroida sellaiseksi, että Monsterit pystyisivät kulkemaan Tile-luokan ilmentymien perusteella. Sen avulla reittiä olisi mahdollista muokata myös tiedostosta. Tämä edellyttäisi kuitenkin kaikkien Pathiin viittaavien luokkien toiminnan refaktoroimista. Jatkokehityksen kannalta olennainen kohde.



Kuten projektista tehdystä UML-kaaviossa huomaa, useat luokat ovat tietoisia toisistaan. Tätä voitaisiin ehkäistä luokkien rajapintoja kehittämällä. Tämä kuitenkin veisi aikaa ja uudistusten myötä myös rajapinnat pitäisi päivittää. Näin alkuvaiheen kehitys hidastuisi. Vaikka UML hieman hirvittääkin, vähentää luokkien tietoisuus toisistaan silti kirjoitettavan ohjelmakoodin määrää huomattavasti. Vaikka osassa kohdista ohitankin rajapintojen käytön, se ei tarkoita sitä, etteikö niitä kuitenkin ole ja hyödynnetä. Muun muassa GUI:n ja Worldin sekä Worldin ja LevelLoaderin kommunikointi tapahtuu pääasiassa rajapintojen kautta.

5. Algoritmit

Projektissa käytettävät algoritmit perustuvat pääasiassa yksinkertaiseen trigonometriaan. Tower etsii omalla ampuma-alueellaan olevat Monsterit laskemalla hypotenuusan Monsterin ja omien koordinaattiensa välille. Näistä monstereista se valitsee sen, joka on edennyt reitillä pisimmälle. Shoot-metodilla torni laskee Projectilen lentoradan omien ja kohteen koordinaattien perusteella ja arctan:n avulla. Kulma on korjattava, sillä arctan:n Af = [-Pi/2,Pi/2]. Tower antaa kulman luomalleen Projectilelle, joka sen perusteella etenee muuntamalla kulman nopeuden ja sinin sekä kosinin kanssa koordinaattien muutokseksi.

Päädyin ratkaisuihin niiden käytännöllisyyden takia. Mahdollinen kehitysaskel olisi Projectilejen kehittäminen siten, että ne ammuttaisiin Monsterin tulevan lokaation perusteella, eivätkä ne silloin menisi ohi. Sen tekeminen tämänhetkisen Pathin kanssa olisi kuitenkin vaatinut runsaasti enemmän laskemista, joten tyydyin nopeasti vihollisen sijaintia kohti lentävään ammukseen.

6. Tietorakenteet

Merkittävimmät tietorakenteet, joita käytetään tietojen säilömiseen sijaitsevat World- luokassa. Luokan on pidettävä kirjaa kaikista maailmassa olevista ja sinne tulevista olioista. Käytän tiedon tallennukseen yksinomaan muuttuvatilaisia kokoelmia, sillä ohjelmaa suoritettaessa samojen kokoelmien alkioita vaihdetaan jatkuvasti. Buffer-kokoelmaa käytän säilömään Monstereita, Towereita sekä kaikkia Entityjä. Toinen merkittävä kokoelmatyyppi on Queue, jota käytän sen käytännöllisyyden vuoksi jonomaista toiminnallisuutta tarvittaessa, kuten tulevien vihollisaaltojen kanssa ja vihollisten ilmestymisjonossa. Outolintu kokoelmissani on ListBuffer, jota käytän Projectilejen säilyttämiseen. Tietorakenne on kriittinen Projectilejen toiminnan kannalta, sillä sitä käytettäessä muut Projectilet eivät pääse vaikuttamaan vihollisen vahingoittamisessa tapahtuvaan laskentaan(lisää alempana).

7. Tiedostot ja verkossa oleva tieto

Yksittäisolio LevelLoader käsittelee tekstitiedostoja. //:lla merkityt alueet ovat dokumentissa selvyyden vuoksi lisättyjä kommentteja. Tieto on esitettävä seuraavasti:

#waves	//header1, vihollisaallot
normal*5:0.5	//1. aalto, 5 tavallista Monsteria 0,5s välein
normal*2,tank*3,fast*4	//2. aalto, 2 tavallista, 3 tankkia, 4 nopeaa 0,2s välein
#stats	//#header2, maailman asetukset
money=1000	//1000 rahaa aloituksessa
health=100	//100 elämäpistettä alussa

Kukin vihollisrivi on oma aaltonsa. Yleinen rakenne

<viholliseen tyyppi>*<lukumäärä>,…,<tyyppi>*<määrä>:<vihollisten ilmestymisnopeus>

Kun virheellisen tiedoston lataa pelin Game menu- valikon kautta, niin ruudulle ilmestyy Alert, joka varoittaa käyttäjää virheellisestä tiedostosta ja ehdottaa mahdollisen korjausehdotuksen. Tiedostossa lukemisessa on bugi, mikäli käyttäjä muokkaa valmiiksi annettua tasoa, joka ladataan automaattisesti siten, että se muuttuu virheelliseksi, niin peli ei käynnistyessään lataa aaltoa. Tiedostosta lukemista voisi kehittää yksikkötestien avulla.

8. Testaus

Ohjelmaa testattiin pääasiassa graafisen käyttöliittymän ja konsolin avulla. Graafisen käyttöliittymän avulla oli helppo selvittää, että maailmassa esiintyvien olioiden perustoiminnot toimivat oikein. Yksittäisiä bugeja korjasin eristämällä konsolin avulla virheen sijainnin ja korjaamalla rikki olevan kohdan. Tämä vastasi suunnitelmassani suunnittelemaa testausta. Tiedostosta luku olisi voinut myös testata yksikkötestien avulla.

9. Tunnetut puutteet ja viat

- Äänitiedostot soivat viiveellä. Muokkasin Audacityä käyttäen äänen alkamaan välittömästi toistettaessa. Kuitenkaan ne eivät aina soi ajallaan. Tässä voisi hyödyntää kenties rinnakkaisohjelmointia. En ole kuitenkaan varma kuinka tehokkaasti sillä Audio-luokan ilmentymät play():ta kutsuttaessa pyörinevät automaattisesti erillisessä säikeessä.
- Projectilet voivat mennä huti; bug or feature?
- Jos alkuperäistä tasotiedostoa muuttaa, niin peli voi käynnistyä ilman tasoa, mutta ei ilmoita virheellisestä tiedostosta. Korjaus mahdollista kovakoodaamalla perustaso tai pakottamalla käyttäjän lataamaan tason tiedostosta. Erillinen aloitusnäyttö olisi hyvä ratkaisu.
- Projectile-luokka aiheuttaa virheen World:n updatessa mikäli niitä säilövä kokoelma ei ole ListBuffer. En osaa tarkalleen sanoa onko kyse ListBufferin ominaisuuksista. Tapausta lienee hyvä tutkia. Mahdollinen virhe aiheutuu luullakseni silmukan rinnakkaisesta suorituksesta, jollain saman Monsterin samalla hetkellä poistavista projectileista toinen aiheuttaa virheen, monsterin puuttuessa kokoelmasta. Kokoelman tarkistaminen ennen monsterin poistamista poistaisi virheen, mutta aiheuttaisi uuden bugin, sillä toinen projectileista katoaisi vihollista vahingoittamatta.
- Jos tornin asettaa juuri oikeaan kulmaan, niin sen voi saada ampumaan 180 astetta väärään suuntaan. Tapahtuu hyvin harvoin ja johtuu luultavasti laskentatarkkuudesta ja -tavasta. Voitaisiin luultavasti korjata ainakin johonkin asti muuttamalla tapaa, jolla Tower ampuu. Ei olennaisesti häiritse peliä.

10. 3 parasta ja heikointa puolta

Hyvät:

1. Projekti on hyvin pelattavissa oleva kokonaisuus. Tasojen muokkaus on helppoa ja pelissä saa aikaan kiinnostaviakin tasoja.

2. ScalaFX-grafiikkakirjaston käyttäminen. Vaikka peli ei graafiselta asulta niin loistokas olekaan, niin kokonaisuus on silti toimiva. Onnistuin käyttämään ScalaFX:n tarjoamia tyylikkäitä luokkia kuten FileChooseria ja Alertia mielekkäästi. Kirjastovalinta ja sen käyttö onnistuivat
3. Ylipäätään kokonaisuus on suhteellisen selkeä ja hyvin muokattavissa. Toiminnot pyörivät korkeamman asteen metodien avulla. Projektia on helppo jatkokehittää.

Huonot:

1. Pelin sisältö jää hieman suppeaksi. Tornityyppejä on vähän ja mekaniikat yksipuoleiset. Pelin kiinnostavuus ei säily kovin pitkään
2. Pathin toteutus rajoitti pelin mahdollisuuksia. Tulevaisuudessa toteuttaisin sen eri tavalla. Samaan liittyen myös Projectilejen toiminta pitäisi uusia
3. Selkeys koodissa on yhä paljon turhia ja turhan monimutkaisia kohtia, joiden kehittäminen tekisi hyvää.

11. Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Ensiksi toteutin karkean graafisen käyttöliittymän, pelin olioiden luomisen pohjalle. Tein World:sta alustavan luokan ja aloitin Towerin, Monsterin, ja Projectilen luomisen. Loin Pathin Monstereille. Sen jälkeen aloin täydentää luokkia ja käyttöliittymää. Lopuksi tein LevelLoader-luokan ja viimeistelin grafiikat sekä äänet.

Alkuperäinen suunnitelmani, johon punaisella merkitty toteutuneet päivämäärät:

18.2-3.3 15.3	• ScalaFX:ään tutustuminen • Pelin malliin vaadittavien luokkien alkeellisten versioiden rakentaminen • Käyttöliittymän rakentamisen aloittaminen
4.-17.3 28.3	• Käyttöliittymän kehittäminen • Mallin luokkien kehittäminen • Luokkien testaus
18.-31.3 12.4	• Käyttöliittymän kehittäminen • Luokkien kehittäminen toimiviksi • LataaPeli-luokan kehittäminen • Uusien aliluokkien kehittäminen piirreluokille Vihollinen ja Torni • Luokkien toiminnan testaus
1.-14.4 24.4	• TallennaPeli-luokan kehittäminen • Grafiikoiden ja äänien suunnittelu • Uusien aliluokkien kehittäminen piirreluokille Vihollinen ja Torni

	• Luokkien toiminnan testaus: Pelin pitäisi olla pelattava ja toimintakunnossa
15.-23.4 28.4	• Pelin viimeistely grafiikoiden, äänien ja toiminnan osalta

Olin lähestulkoon koko projektin ajan jäljessä aikatauluarviostani. Loppua kohden otin kiinni, mutta en olisi saanut projektia ajoissa tämänhetkiseen kuntoon, joten pyysin lisäaikaa. Etenemisjärjestys muistutti pitkälti suunniteltua.

12. Kokonaisarvio lopputuloksesta

Olen tyytyväinen siihen, mitä sain aikaan. Peli on jo nyt pelattavissa ja tarjoaa hyvän pohjan tulevalle kehitykselle. Rakensin ensimmäistä kertaa itse suuremman ohjelman, jossa hyödynsin kuvaa, ääntä, ulkopuolisia kirjastoja ja sbt:tä. Sain rakennettua toimivan luokkarakenteen, joka pysyi riittävän yksinkertaisena ja mahdollisti jatkuvan muokkauksen ja kehityksen. Toisin kuin tekstipelin kanssa koodin sekavuus ei tullut seinänä vastaan ja estänyt jatkokehitystä. Puutteet johtuvat lähinnä ohjelmoinnin puutteesta. Tein sen, minkä ehdin näissä aikavaatimuksissa ja olen senkin osalta ihan tyytyväinen.

Tulevaisuudessa ohjelmaa voi parantaa: ohjautuvat projectilet, lisää monstereita, monstereiden vahvuus skaalautuu, lisää torneja, torneille kehitysvaihtoehtoja, torneille ja ammuksille erityisvaikutuksia, tekstuurien päivittäminen, äänten päivittäminen, Pathin toimiminen gridiin perustuen, Level Editor, wave editor, waven spawn raten muuttaminen kesken waven, aloitusnäyttö, lopetusnäyttöjen uudelleentekstursointi, tornien lisääminen painettavista kuvista eikä valikoista, tornit näyttämään rangen kun niitä klikkaa, tornin ostettua ns. haamutorni seuraa rangen kanssa ennen kuin sen asetta, säännöt tornien asettamisen kanssa: ei toistensa päälle, ei tien päälle, tutoriaali, cutscenet, musiikki taustalle ja pari muuta.

Olen suhteellisen tyytyväinen tietorakenteisiin, jotka olen valinnut. Tähän asti ne ovat tarjonneet sujuvan ohjelman suorituksen. Skaalautuvuutta voi tarkastella paremmin tulevaisuudessa, mutta tämänhetkisiin olosuhteisiin ne riittävät hyvin. Uskon, että ohjelman rakenne soveltuu hyvin muutosten tekemiseen tulevaisuudessa.

Jos saisin nyt valita, mitä tekisin toisin, aloittaisin Pathin tekemisen suoraa Gridin Tilejen perusteella. Näihin aikarajoitteisiin nähden olen suhteellisen tyytyväinen aikaansaamaani lopputulokseen.

13. Viitteet

Ohjelmointistudio 2:n verkkomateriaali: https://plus.cs.hut.fi/studio_2/k2019/toc/

Scala Jesus Mark Lewis: https://www.youtube.com/channel/UCEvjiWkK2BoIH819T-buioQ

ScalaFX GUI Library: http://www.scalafx.org/

Audacity: https://www.audacityteam.org/

(Äänet tehty itse)