OHJELMOINTIPROJEKTI OHJ3: GENOCIDAL NYSSEMAN

Aki Kankaanpää ja Samuli Käppi

JOHDANTO

Projektimme Genocidal Nysseman, hyvin kaukaisesti Pacmanin tyyppinen spoofpeli, jonka tarkoitus on viihdyttää sen käyttäjäänsä hetki, kunnes he kyllästyvät sen täysin arpomalla muodostamaan pohjaan sekä silmiä häiritsevän pihatonttuneliön pikaisiin liikkeisiin.

PELIN SELITYS

Genocidal Nysseman on labyrinttiseikkailu urbaanissa koronansyöksemän Tampereen keskustassa, jossa renkaan alle jää maskittomat Jeret, poliisit, voimistajat sekä harvoin kerroin mystinen puutarhatonttu, ja kaikki tiet ovat luotisuoria ja 90 asteen kulmissa kuin viivoittimella olisi mitattu, kaikkien muiden teiden joko mysteerisesti hävinneen tai ollessa ratikkatyömaan koukereissa. Pelin päätarkoitus on pitää bussikuskin raivo päällä, bussi täynnä terveitä, pelastettuja sieluja ja koronankourissa olevat maskistakieltäytyjät maantasolla. Pelissä kontrolloit hyvin neliömäistä ja teräväkulmaista bussia, maalattuna pelottavan mustaksi joka puolelta, ja tarkoituksenasi on keräillä erinäisiä, nekin hyvin teräväkulmaisia ja neliömäisiä objekteja.

Pelin käynnistyessä, sinut ottaa vastaan kaksi ikkunaa suurempi ikkuna nimettynä Genocidal Nysseman, jossa itse peli tulee olemaan, sekä pienempi ikkuna Setup Window, josta voit valita pelissä olevan vaikeusasteen ja haluamasi nimen. Tämän ikkunan sulkiessasi rastia painamalla tai vaihtoehtoisesti cancel-napista, sammuu koko ohjelma iso ikkuna mukaanlukien. Painaessasi Start-nappulaa ja ollessasi syöttäneen jonkinlaisen merkkijonon syöttökenttään, peli alkaa pikimiten isossa ikkunassa, sammuttaen pienin ikkunan. Jos unohdat syöttää merkkijonon, saat pelin sijasta pienin popup-ikkunan, joka muistuttaa nimen valitsemisesta, jonka voi sulkea haluamallaan tavalla. Pelin alkaessa, kartalle ilmaantuu huomattava määrä erinäisiä neliönmuotoisia objekteja muutamissa eri väreissä, joita sinun kuuluu kerätä haluamassasi järjestyksessä, ja bussi ei odottele, lähtien suoraan vasemmasta yläreunasta oikeallepäin.

Peli loppuu muutamalla eri tavalla; joko voitat keräämällä kaikki Maskinkäyttäjät ja yliajamalla kaikki Maskistakieltäytyjät, tai saamalla kiinni mystisen puutarhatontun, joka karkaa sinulta nopeammin kuin keskiverto koronapotilas sairaalasta lähimmän hautasmaan nummelle; tai häviät joko ajamalla seinään tai nukahtamalla ikuisiksi ajoiksi koronan lempeään kämmeneen ainoan sinua elossa pitävän asian, raivon, loppuessa kesken. Aloitat pelin täydellä raivolla, jota on yhteensä 900, ja se kuluu vaikeusasteesta riippuen kokoajan passiivisesti pois, tämänhetkisen raivon määrän ollessa nähtävissä punaisena palkkina vasemmassa yläkulmassa.

Pelissä on 4 vaikeusastetta: Extremely Easy, Easy, Not Easy, sekä Uneasy. Ne vaikuttavat prosentuaalisiin mahdollisuuksiin, kuinka usein tietynlaisia gamepiecejä lisätään kentälle, sekä pelin kellon nopeuteen. Ainoat objektit jotka ovat tältä immuuneja ovat pelaajan bussi ja mystinen tonttu. Alla on nopea selitys näistä vaikeusasteista (jokainen prosentuaali on muodossa ”% mahdollisuus syntyä per ruutu”, ja ruuduksi lasketaa yksi bussin kokoinen alue joka on 10x10 pikseliä suuri):

Vaikeusaste	Vihreä	Punainen	Sininen	Keltainen	Ruutuja / sec
Extremely Easy	5	6	2	2	4
Easy	5	5	2	1	5
Not Easy	5	5	1	1	5,88...
Uneasy	0	0	0	10	7,14

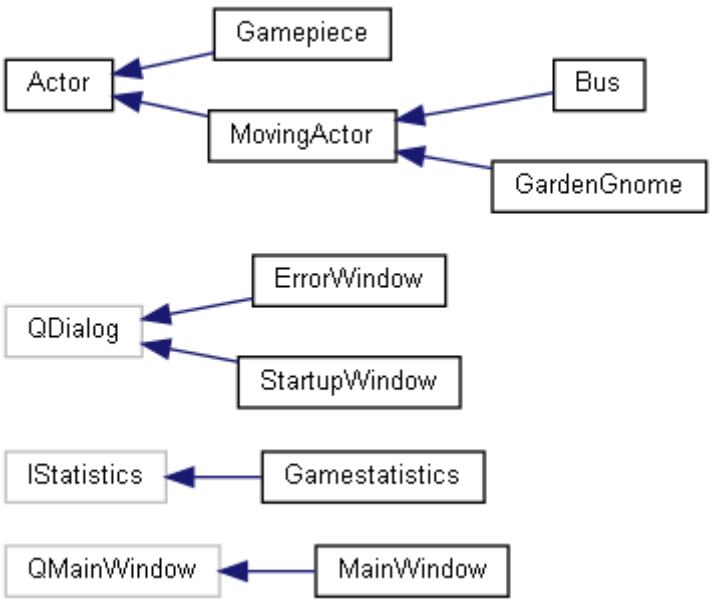
Pelissä olevat ”toimijat” ja niiden effektit on selitetty alla:

- Maskinkäyttäjä (vihreä): Tämän objektin kerätessäsi saat yhden matkustajan kyytiisi, menetät -50 raivoa, mutta jos onnistuneesti voitat pelissä, saat 1000 pistettä per matkustaja joka on yhä elossa pelin päättyessä.
- Maskistaieltäytyjä (punainen): Tämän objektin kerätessäsi saat 100 pistetää sekä 200 raivoa, mutta aina kun ajat Maskista Kieltäytyjän päältä, jokaisella kyydissä olevalla Maskinkäyttäjällä on 30 % mahdollisuus saada korona ja joutua sairaalaan, menettäen tämän matkustajan bussistasi.
- Tonttu (magenta): Tämän objektin kerätessäsi voitat pelin automaattisesti ja saat 7777 pistettä, sekä saat salaisen voittamisruudun näkyviin. Tonttu imitoi aina parhaansa mukaan sinua, menemällä aina samansuuntaisesti kanssasi riippumatta sen sijainnista, tosin jos seinä tulee vastaan tontun pasman menevät sekaisin ja se saattaa joko seisoa paikallaan tai liikkua satunnaisesti johonkin suuntaan.
- Poliisi (sininen): Tämän objektin kerätessäsi menetät 250 pistettä, mutta saat 450 raivoa takaisin, tämän bussikuskin vihatessa virkavaltaa yli kaiken.
- Pelaaja (musta): Sinä itse liikutat tätä hahmoa käyttämällä WASD-näppäimiä. Risteyksissä ei ole pakko olla pikselitarkka, kunhan painat hieman ennen käännöstä mihin haluat kääntyä niin kyllä bussin laaja kääntymiskaari hoitaa loput. Tämä kääntymiskaari kannattaa pitää mielessä, koska bussi ei kykene tekemään 180 asteen käännöstä kolikon päällä eikä pelissäkään.
- Voimanantaja (keltainen): Tämän objektin löytäessäsi saat et vain huomattavan 500 pistettä, vaan myös pysyvän bonuksen kunnes poimit toisen Voimanantajan maasta. Nämä bonukset ovat: raivo ei kulu passiivisesti (RAGE_IMMUNITY), pisteiden tuplaus (POINTS_MULTIPLIER), sekä maskistakieltäytyjiä kerätessä immuniteetti maskinkäyttäjien sairastumiseen ja menettämiseen (DISEASE_IMMUNITY).

Muut huomattavat pelifaktat:

- Keltamustat raidatut alueet ovat tällähetkellä ratikkatyömaan alla ja niissä olevia katuja ei tällä hetkellä voi käyttää.
- Pelin ei ollessa käynnissä, voit painaa r-näppäintä ja palauttaa aloitus-popup ikkunan, josta voit aloittaa uuden pelin sekä samalla vaihtaa valittua nimeäsi sekä vaikeusastetta.

OHJELMAN RAKENNEKAAVIO



SOPIMUSSUUNNITTELU

Alla on näkyvissä meidän käyttämämme luokkien esi- sekä jälkiehdot:

1) Actor: gametoken olisi hyvä olla asetettuna gamesceneen, sekä koordinaatien pitäisi olla oikeaa tyyppiä sekä kattavat pelin ajamisen takia. Tämä mahdollistaa helpon QGraphicsRectItem gametokenin paikan tarkistuksen ja sen sallittujen ruutujen tarkistuksen.
2) Bus: setCurrentPowerup arvon antaessa metodille sen tulisi olla välillä 0-3. Bus pystyy tällöin myöhemmin palauttamaan tämän arvon, jotta powerupin senhetkinen tila on mahdollista tarkistaa tälle bus-objektille.
3) Errorwindow: Ei huomioitavaa.
4) Gamepiece: Constructorissa arvo type on oltava välillä 0-3. Tällä arvolla on huomattavasti helpompi saada selville mikä tietyn Gamepiece objektin tyyppi on, käyttämällä returnPIecetype() metodia.
5) Gamestatistics: setGameDifficulty pitää antaa sopiva difficulty-arvo välillä 0-3. Tällä vaikeusasteella Gamestatistics pystyy ylläpitämään passiivista raivon vähenemistä rageDecay metodilla, vaikeusasteen määrätessä tämän raivon passiivisen menetyksen määrän. Jos sille annetaan liian suuri arvo raivoo (esimerkkinä 10000), sekä osaa automaattisesti muuttaa sen takaisin maksimiin eli 900:n, ja sama pätee myös alaspäin mennessä.
6) Gardengnome: Kunhan pelaajan suunta on aina oikeanlainen (0-3), pitäisi gnomen kykeneä aina yrittämään liikkua samansuuntaisesti pelaajan kanssa tai yrittämään satunnaisesti liikkua johonkin suuntaan, jos pelaajan valitsema suunta ei ole mahdollinen.
7) Mainwindow: Kunhan mainwindowille annetaan oikeanlaisia arvoja aluksi (eli Qstringi ja difficulty), kykenee se yhteistyössä GameStatisticsin kanssa (jonka se luo näistä arvoista riippumatta) hallinnoimaan peliä, palauttamaan haluttuja arvoja, sekä päivittämään arvoja sen kyljessä tai uudelleenkäynnistämään pelin sekä samalla joitain sen ja Gamestatisticsin sisältämiä arvoja samalla, kunhan arvot Gamestatisticsissa sekä sen muutamassa eri muuttujassa (tärkeimpinä list_of_gamepieces_ sekä legal_coordinates_), kunhan näiden arvot pysyvät oikeanlaisina ja ne luodaan oikeaan muotoon jo alussa.
8) MovingActor: Antaessa direction-arvoja, on arvojen oltava välillä 0-3 ja liikkumismäärän amount olisi hyvä olla 1. Mainwindow ei kykene antamaan vääriä arvoja, paitsi jos tiedosto tamperecoordinates.txt on viallinen. MovingActor kykenee liikuttamaan tätä QGraphicsRectItemiä, kunhan sille on annettu oikeaan sceneen alustettu RectItemi sekä oikeantyyppiset koordinaatit.
9) Startupwindow: ottaa käyttäjältä inputtina merkkijonon sekä heidän päättämänsä vaikeusasteen. Kunhan merkkijono ei ole tyhjä, startupwindow mahdollistaa pelin aloittamisen ok-napin painalluksella, sekä erinäisten vaikeusasteesta kiinniolevien muuttujien alustamiseen.

OHJELMAN TOIMINTA

Ohjelma käynnistyy avaamalla sekä mainwindow että startupwindow objektit. Mainwindowin constructorissa tapahtuu paljon asioita heti alkuun, kuten kuvien pixmappaaminen gameimages_ mappiin, startupwindowin signaalien yhdistäminen mainwindowin slotteihin, Gamestatistics objektin alustaminen gamestats_:iin, sekä erinäisten pelialueiden sekä random seedin srand(time(0)) alustaminen (ragemeter_ ja ragescene_, gamescene_ ja sen pixmap kuva). Startupwindowista voi valita nimen sekä vaikeusasteen, ja kun nimi on valittu siten, että se on minkäänlainen merkkijono, se emittaa difficultySignal nimisen signaalin, joka aloittaa pelin createGamella. Jos sensijaan pienen tai suuren ikkunan sulkee, molemmat sulkeutuvat, parenthoodin tai signaalin rejected kautta. Pelin alkaessa ensimmäistä kertaa, createGame slotti luo: readCoordinatesilla kartan, lukien rivejä tamperecoordinates.txt tiedostosta ja prosessoi sen insertCoordinates metodilla legal_coordinatesiin, muodossa <int,<vector<int>>, jossa ensimmäinen integeri on x:n arvo, ja vektorissa on kaikki käyvät y:n arvot sillä x-sarakkeella; Pelaajan ja Tontun QGraphicsRectItemiin pointterin ja antaa ne gnomerect ja playertoken muuttujille, jotka annetaan siitä vielä pidemmälle player_ ja gnomelle_. Se asettaa kaikki labelien arvot, jotka muuttuvat pelin aikana ja ovat näkyvissä oikeanpuoleisissa labeleissa, nollaksi (kun pelin resettaa, kaikkia arvoja ei palauteta defaulttiin). Tämän jälkeen, se kutsuu metodia setDifficulty, joka: luo pelin gamepiece objektit käyttämällä rand()ia sekä createGamelle annettua vaikeusastetta parametrinä käynnistäessä, päättää ja käynnistää pelin sisäisen kellon tick_timerin_, jonka nopeus riippuu vaikeusasteesta, sekä tallentaa tällähetkellä olevien maskinkäyttäjien ja maskistakieltätyjien määrän käyttämällä resetRemainingPedestrians metodia Gamestatisticsille. Peli tekee vielä muutaman arvon resettaamisen, kuten lcd ruudun, ja sen jälkeen peli alkaa

Bus objektin liikkuessa vasemmasta yläkulmasta oikeallepäin. Pelin kulku on seuraava: pelaaja antaa näppäimenpainalluksella (WASD) arvon talteen queued_directioniin. Kun timerin current_tick saavuttaa 10 (jokaisella tickillä liikutaan yksi pikseli, tonttu liikkuu 10, sekä vähennetään passiivisesti raivoa), jota nostetaan slotissa tickHandler timeoutin mukaisesti, tick resetataan, tarkistetaan voiko bussi liikkua suuntaan verraten sen tulevaa x,y pistettä suunnan perusteella haettuna legalCoordinatesin x-avaimen y-vektorista tarkistaen, onko arvo löydettävissä. Jos on, voidaan liikkua, muulloin peli loppuu törmäykseen. Jos kohdassa on gamepiece-tyyppinen objekti list_of_gamepiecessissä, checkGamepieceCollision siirtää kaikki tällaisien arvojen indeksit vektoriin, tekee niillä määritellyt asiat ja muuttaa arvoja, poistaa ne vektorista ja muistista, vähentäen currentPedestrians muuttujaa Gamestatisticsluokassa jos relevanttia. Jos huomataan että currentPedestrians on 0, peli loppuu voittoon. Muita lopetuksia ovat jos ticktimerin tai kerätyn pedestrianin takia raivo menee alle nollan, tai jos pelaaja saa tontun kiinni juuri oikealla hetkellä että se on heidän seuraavassa ruudussaan tarkalleen. Kutsutaan endGame metodia gamestate conditionilla, joka määrittää mihin peli loppui, asetetaan kaiken päälle pixmappi riippuen pelin loppumistavasta, pysäytetään tick_timer_, nostetaan pelattujen pelien ja voittojen / häviöiden määrää, ja tyhjennetään list of gamepieces arvoistaan ja poistetaan kaikki paitsi itse vektori. Pelin voi tällöin resettaa r-näppäimellä, jolloin peli resettaa muitakin arvoja, poistaa ja uudelleentekee esimerkiksi player_ ja gnome_muuttujan, mennen eri haaraan createGamessa nyt kun pelatut pelit (totalnysses joka saadaan returnTotalNysses metodilla gamestats_:ilta) on yli nolla, ja aloittaen pelin alusta. Pelin sammuessa, poistetaan kaikki poitterit sekä objektit destructoreilla, ja ohjelma sammuu.

VAPAAEHTOISET OMINAISUUDET

1) Oma: Kartan ruudukon ja koordinaattien toteutus. Kartan ruudukko on luotu käyttäen .txt tiedostoa, jossa on rivejä, jotka ovat muotoa x:y1,y2,y3... yn. Näistä arvoista luodaan tyyppiä map<int,vector<int>> tyyppinen local_coordinates_ muuttuja, joka ylläpitää kaikkia vaadittuja koordinaatteja, x:n mukaan määritettynä. Jos halutaan tarkistaa esimerkiksi mitkä ruudut ovat laillisia liikkumista varten, täytyy vain tietää tämänhetkinen liikesuunta, josta voidaan selvittää tulevan x:n arvo, joka toimii avaimena, ja mennä läpi sen arvona mapissa olevan vektorin sisällä olevat arvot läpi, säästäen huomattavasti työtä kaikkien läpikäynnin sijaan.
2) Listassa: Tasainen ruudunpäivitys. Kaikki liike, mitä ruudulla tapahtuu, on toteutettu Qtimer tyyppisellä objektilla, ja jokaisella timeoutilla liikuttaa pelaajaa sekä tonttua. Pelaaja voi vaihtaa suuntaansa vain joka kymmenes ”tickki”, tätä ylläpitäen muuttuja current_tick_, jonka kymmenennellä tickillä tarkistetaan mihin suuntaan seuraavaksi liikutaan, voiko siihen mennä (käyttäen kohdan 1 kartoitusta), ja tarkistaen onko siinä jotain Gamepiece tyyppistä entiteettiä, jonta effekti voitaisiin aiheuttaa. Tonttu sensijaan pomppii ympäri kenttää joka tickillä, liikkuen käytännössä 10 kertaa nopeammin kuin pelaaja, mutta käyttämällä strategiaa on se mahdollista saada kiinni, nähty ja koettu.
3) Listassa: Pelihahmon tasainen liike. Jatkaen kohdasta kaksi, pelaajan hahmo liikkuu tasaisesti valittuun suuntaan Qtimer ajastimen mukaisesti, kunnes se törmää, sen raivo loppuu, tai pelaaja päättää vaihtaa sen suuntaa. Tämä liikkeen suunta on yllä Bussilla, periytyneenä MovingActorilta, ja sitä voidaan vaihtaa myös MovingActorin metodilla setDirection(). Nopeuteen voi vaikuttaa pelin vaikeusasteen kautta, vaikeampien vaikeusasteiden nopeuttaessa sekä bussin että tontun nopeutta. Tontun liike ei ole yhtä tasaista, mutta se oli haluttu ominaisuus.
4) Listassa: Pelihahmon päivitykset. Kerätessään voimanantajan, saa pelaaja pysyvän parannuksen bussiinsa, satunnaisesti kolmesta vaihtoehdosta, joiden nimet on kuvattu Bus.h:ssa enum powerup alla, ja johon se myös tallentuu current_powerup_ muuttujaan. Niiden effekti on asetettu ympäri koodia kun se on relevanttia, esimerkiksi mainwindow.cpp rivillä 306 RAGE_IMMUNITY parantajan suhteen.
5) Listassa: Pelin tilan seuranta. Pelin aika kerättävistä tiedoista, pelaajalle on aina näkyvissä peliruudun oikealla puolella tällä hetkellä olevan tämän pelin pisteet (lcd), kyydissä olevien matkustajien määrä, jäljellä olevien maskinkäyttäjien ja maskikieltäytyjien yhteinen määrä pelin loppuun, tämän pelisession kokonaispistemäärä sekä kokonaispelimäärä, niistä hävityt ja voitetut pelit, tämänhetkinen voimanantaja (alussa none) (label) sekä raivon tämänhetkinen määrä punaisena palkkina (scene).
6) Oma: Reset-hotkey. Pelissä on toteutettuna reset-hotkey, jolla voi pelin ei ollessa käynnissä palauttaa tälle operaatiolle relevantit arvot, mutta säilyttää muistissa arvot joita ei ole syytä hukata, kuten esimerkiksi pelinaikana ylläpidetty pelimäärä, voittosuhde sekä yleispisteet.

TEKIJÖIDEN VASTUUJAKO

Projektin vastuunjako oli kohtuullisen tasainen molempipuolisesti, johtuen hyvin myöhäisestä aloituspäivämäärästä, mutta jako meni kutakuinkin näin:

Aki

- Actor, MovingActor, Gamepiece, Bus, sekä Errorwindow ja StartupWindow classit sekä niiden sisältämät metodit, sekä kaikki ohjelmasta löydettävät enumit, sekä joitakin määritelmiä
- Actor ja siltä periytyvät luokat sekä niiden oliosuunnittelu, johon kuuluu periytettävät luokat Actor ja MovingActor sekä enemmän funktionaaliset Gamepiece ja Bus luokat sekä GardenGnome luokan rajapinta
- Yksikkötestien toteuttaminen sekä testaaminen Gamestatistics-luokalle, sekä Gamestatistics-luokalle IStatistics-rajapinnan toteuttaminen ja sen virtuaalisien funktioiden yksikkötestaus
- Resurssitiedostojen yhdistäminen projektiin
- Kaikkien .h-tyyppisten tiedostojen kommentointi, sekä suurin osa .cpp-tiedostoista lukuunottamatta mainwindow.cpp, gamestatistics.cpp sekä tst_gamestatistics.cpp
- Yksinkertaiset ikkunoiden väritykset sekä niissä olevien objektien paikantaminen, asettelu sekä suunnittelu
- Lisäominaisuuksista tasainen ruudunpäivitys, tasainen liike sekä tilan seurannan alkudraftit
- Tämän dokumentin kirjoittaminen

Samuli

- Mainwindow.cpp:ssä toteutetut algoritmit sekä olio GardenGnome
- Muistinhallinta lähestulkoon koko projektiin muutamia lisäyksiä lukuunottamatta, käyttäen etäyhteydellä Tunin linux-palvelimilta valgrindia
- Erinäisten karttakuvien, pelinlopetuskuvien sekä muut kuvitusta lisäävät ominaisuudet sekä tiedostot
- Mainwindow.cpp, gamestatistics.cpp sekä tst_gamestatistics.cpp tiedostojen kommentointi (vähemmän tiedostoja, paljon enemmän kommentoitavaa)
- Lisäominaisuuksista pelihahmon päivitykset, tilan seurannan viimeistely sekä itsetoteutettu reset-nappula
- Projektin vihdoinviimein aikaansaaminen sekä tehtävän kartoittaminen
- Loppusiistiminen, pelin pelattavuuden parantaminen pelimekaaniselta tasolta
- Kuvien tuottaminen tähän dokumenttiin

TIEDOSSA OLEVAT ONGELMAT / PUUTTEET

- Kysytyllä nimellä ei tehdä mitään
- Highscore-taulua ei saatu toimimaan
- Tarjoamia rajapintoja ei juuri käytetty