ELEC-A7150 - C++ Programming, Group: Angry Birds 2

Aku Viitanen 481548 - Project Manager
Konsta Varonen 432458
Charlotta Alén 528375
Ilkka Malassu 430463

1. Peli yleisesti

Pelin toteutus muuttui suunnitellusta melkoisen paljon aikarajoitteiden ja käytettyjen kirjastojen vaatimusten vuoksi. Peli sisältää keskeisimmät komponentit, mutta osa tukevista järjestelmistä kuten esimerkiksi valikot uupuvat vielä. Itse varsinaisessa pelissä kaikki osat fysiikoista pisteiden laskuun toimivat tekniikan kannalta, mutta niiden arvot ja pelin säännöt eivät ole valmiita.

Pelistä löytyy kirjoituksen hetkellä 3 .csv-tiedostoista ladattua kenttää, yksi lintu erikoisominaisuudella, yhdenlaisia esteitä sekä maalitauluja, taustamusiikkia ja yksi ääniefekti linnulle. Lisäksi peli laskee pisteitä osumien mukaan ja kenttä loppuu kun pelaajalta loppuvat linnut tai kaikki kohteet on tuhottu.

Peli lataa kentät tällä hetkellä järjestyksessä, tosin kenttien lataus .csv-tiedostoista tehdään kutsumalla worldbuilder-funktiota tiedoston nimellä, joten kenttävalinnan luonti ei olisi kovin monimutkainen homma.

2. Ohjelman rakenne

Alkuperäisestä suunnitelmasta poiketen pudotimme statemachinen käytön pois projektista, koska emme kokeneet sen tuovan hyödyllisiä ominaisuuksia tässä yhteydessä. Samoin siirryimme fysiikoiden osalta käyttämään Box2D:tä.

Itsetehtyjä luokkia projektissa on viisi, joista kolme kuvaa pelin objekteja ja kaksi fysiikkamalleja. Objekteja kuvaavat Box, Bird ja Pig. Näiden luokkien tehtävänä on piirtää objektit fysiikkamalleilta saatujen parametrien mukaan, sekä objekteille pelilogiikan kannalta tärkeitä tietoja, kuten esimerkiksi maalien hp määriä. Fysiikkaluokkia edustavat RectBody ja CircleBody, jotka on luotu Box2D ohjeistuksen mukaan kuvaamaan nelikulmion ja ympyrän fysiikkamallinnusta. Luokat eivät keskustele lainkaan keskenään, vaan pelin koodi kutsuu itsenäisesti pyörivältä Box2D:ltä fysiikkamallinusten paikan ja rotaation, jonka jälkeen se välittää ne objekteille. Tämä johtuu siitä syystä että SFML ja Box2D omaavat omat esityksensä vektoreille, joten matkan varrella täytyy tehdä tyyppimuunnoksia käsin.

Projektia on jaettu eri tiedostoihin koodin selkeyden vuoksi, tosin itse emopelissä on vielä paljon koodia jonka voisi niputtaa erillisiin funktioihin, tosin koodissa ei ole paljon toistuvuutta, joten se ei tiedostokokoja ja tehokkuutta paljoa nostaisi.

3. Ohjelman logiikka

Ohjelma kutsuu muita luokkia ja ohjelman osia hyvin paljon pääsilmukassaan, jossa se muuttaa esimerkiksi tietotyyppejä kirjastojen käyttöä varten. Pääsilmukassa toteutaan itse pelin varsinainen logiikka, jonka perusteella kutsutaan eri osia esimerkiksi tiedon hakua tai kentän muodostamista varten. Suuri osa pelinaikaisista kutsuista tehdään käyttäen SFML:n ja Box2D:n sisäänrakennettuja ominaisuuksia. Nämä ovat niin kattavia että omia ratkaisuja fysiikan ja piirron kannalta ei ole tarvinnut toteuttaa, vaan silmukassa voidaan keskittyä nimenomaan logiikkaan. Omat parametrit objekteille on tallennettu objekteille, fysiikkamallinukset on jätetty vain Box2D:tä varten, jottei fysiikan simulointi kärsisi.

Yksinkertaistaen tieto liikkuu järjestyksessä Box2D(fysiikka) → pääsilmukka(pelilogiikka) → SFML(piirto ja esitys)

4. Asennus ja käyttö

Ohjelman kääntäminen onnistuu pelkällä make komennolla, makefile hoitaa kaiken käyttäjän puolesta. Lisäksi peli vaatii toimiakseen SFML 2.3 tai uudemman sekä Box2D:n, jonka vaikeahkon asentamisen helpottamiseksi on luotu skripti Installation kansioon.

Ohjelman ajaminen ei vaadi mitään erikoistoimenpiteitä ja peli käynnistyy suoraan.

5. Testaus

Testauksessa käytettiin hyvin paljon konsolia hyväksi. Koodiin laitettiin eri kohtiin cout komentoja, joiden tulosteella testattiin mitä arvoja muuttujat saivat tai pääsikö suoritus koskaan jostain ehdosta läpi. Varsinkin fysiikan testauksessa mitattiin paljon lukuarvoja, koska graafinen toteutus tuli vasta mukaan kun fysiikat oli saatu toimimaan.

Koska ohjelmassa käytettiin paljon luokkia ja niiden kautta referenssejä, muodostui muistin kanssa virheitä hyvin usein. Nämä saatiin kuitenkin helposti metsästettyä valgrindin avulla ja projektin lopullinen versio ei ole testatessa tuottanut yhtään valgrind virhettä.

Kun ohjelmaa käännettiin -Wall -Wextra parametreillä, nousi esiin muutamia käyttämättömiä muuttujia, jotka on sitemmin poistettu, ja SFML:stä johtuvia enumeraatiovirheitä. SFML:n virheitä ei saatu poistettua, tosin ne eivät ohjelman suorittamisen kannalta ole merkittäviä.

Yhteenvetona ohjelmassa ei kirjoitushetkellä ole ilmennyt merkittäviä virheitä millään testitavalla.

6. Ryhmän työskentely

Projektiin osallistuneiden henkilöiden henkilökohtaisten kiireiden takia emme valitettavasti saaneet säännöllistä aikataulua toimimaan, joten toteutus keskittyi muutamalle kokopäiväiselle kerralle. Ensimäisellä kerralla luotiin ohjelman pohjarakenne ja suunniteltiin luokkien rakennetta. Tämän jälkeen jokaista osa-aluetta tehtiin hieman itsenäisemmin. Näihin välitoteutuksiin kuuluivat mm. ulkoisten kirjastojen implementointi, csv-tiedoston lukija, grafiikoiden piirtäminen ruudulle sekä luokkien pohjamallit.

Toisella ryhmäkerralla tehtiin leijonaosa työstä, jonka aikana irralliset osat laitettiin toimimaan keskenään. Grafiikat piirretään fysiikkamallien mukaan, worldbuilder luo pelikentän luokkia ja csv-lukijaa käyttäen. Lopuksi vielä lisättiin pieniä ominaisuuksia, kuten äänentoisto ja pisteidenlasku.

Kaikkia tehtäviä tehtiin ristiin varsinkin niillä kerroilla kun olimme samassa tilassa.
Päävastuualueita oli kuitenkin:
Aku Viitanen: Fysiikka, luokat
Charlotta Alén: Levelreader, grafiikka
Ilkka Malassu: Worldbuilder, grafiikka
Konsta Varonen: Ohjelman kokonaistoiminta, statemachine

Tuntimäärällisesti ryhmän jäsenet osallistuivat työskentelyyn yhtä paljon, mutta koska teimme todella suuren osan työstä lokaalisti ja koneet vaihtoivat käyttäjiä, voivat gitin commit logit antaa hieman väärää kuvaa.