ELEC-A7150 - C++ Programming,

Group: Angry Birds 2

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

1. **Johdanto**
2. **Aikataulu**
3. **Suunnitelma**
   - **3.1. Luokat**
   - **3.2. State machine**
   - **3.3. Fysiikkamoottori**
   - **3.4. Grafiikat**
   - **3.5. Pelin kulku**
4. **Yhteenveto**

1. Johdanto

Suomalainen pelinkehittäjäyhtiö Rovio popularisoi katapulttipelit laajemman yleisön tietouteen vuoden 2009 julkaisullaan Angry Birds. Peliformaatti ei kuitenkaan ole täysin uusi, vaan sen yksinkertaista logiikkaa on nähty hyödynnettävän jo vuosia aikaisemmin. Ohjelmointiprojektina tämän tyyppinen julkaisu on kohtuullisen hyvin jäljiteltävissä. Peliä on kuitenkin tarkoitus operoida pääasiallisesti hiirellä, jonka klikkaus hallitsee täysin ohjelman siirtymiä tilasta toiseen, ja jonka liikkeillä voidaan ohjailla katapultin suuntaa ja jännitystasoa. Tavoitteena on laukaista lintuja erilaisten yksinkertaisten esteobjektien läpi ja kerätä pisteitä osumalla maaleiksi asetettuihin kappaleisiin. Pääsemme projektissa hyvinkin pitkälle perus- ja multimediakirjaston SFML avulla. Muiden kirjastojen hyödyntämisestä päätämme aina tilanteen mukaan. Pelissä olennainen fysiikan mallintaminen on tarkoitus implementoida alusta asti ryhmän jäsenten voimin.

2. Aikataulu

Aikataulullisesti tarkoituksenamme on hyödyntää jäljitelmää ketterän ohjelmistokehityksen mallista, jossa tähtäämme mahdollisimman nopeasti useisiin eri toimiviin demoversioihin lopullisesta ohjelmasta. Ohjelman perusrungon, pelimoottorin ja eri tiloista vastaavan koneiston perusrakenteet ovat suunniteltuina tässä raportissa. Edellä mainittujen osioiden ensimmäisestä versiosta vastaa Varonen ja ne olisi tarkoitus saada toteutettua viikkojen 46 ja 47 aikana. Samoihin aikoihin mahdollista on työstää pelifysiikkaa, josta päävastuussa on Viitanen. Grafiikan piirtämisen ja koodaamisen hoitavat jaetusti Alén ja Malassu. Viikoilla 47 ja 48 visuaalisten ominaisuuksien oletetaan olevan yhdistettävissä pelin perusrunkoon. Yllä jaettujen vastuiden lisäksi kaikki tutustuvat, pohtivat ja auttavat yleisesti kaikilla muillakin projektikehityksen osa-alueilla. Ohjelman viimeistelyn aloitamme viimeistään viikolla 49, ja viimeiseksi jätämme mahdolliset ekstraominaisuudet (esimerkiksi moninpeli), jos ylimääräistä aikaa jää.

3. Suunnitelma

3.1. Luokat

Ylästatet

- onEnter: suoritetaan tilaan siirtyessä
- onUpdate: suoritetaan tilan ollessa käynnissä
- onExit: suoritetaan tilasta poistuttaessa

Main

- Pelin tilat, esim. valikko, peli käynnissä, peli ohi, lataa
- **load**-metodi, lataa .csv-tiedostosta lintujen ja muiden objektien paikat kartalle

inGame

- iGstateMachine: tilojen käsittely
    - waitingState
    - loadingThrowState
    - onAirState
    - pausedState
    - gameEndedState

Engine

- pelin fysiikkatoteutukset: voimat, vektorilaskentaa

Points

- laskee pisteitä **Engine**n triggerien perusteella

Graphics

- Piirtää peliä **Engine**ltä saatavien tietojen perusteella
- Metodit musiikille ja äänitehosteille
- Perii tilat **Engine**ltä, eli esim. **onAirState**ssa keskittää kameran heitettävään lintuun

3.2. State machine

Class inGame

- iGstateMachine
    1. waitingState
    2. loadingThrowState
    3. onAirState
    4. pausedState
    5. gameEndedState

Jokaisella tilalla on **onEnter**-, **onUpdate**- ja **onExit**-metodit. **iGstateMachinessa** on tilan käsittely, jota kutsutaan jokaisen metodin jälkeen. Ennen sitä käsitellään pelin fysiikoihin liittyvät muutokset pelin **Engine**-luokassa ja grafiikat **Graphics**-luokassa. Sen jälkeen kutsutaan oikeata metodia riippuen siitä onko tila muuttunut vai ei. Jos tila on vaihtunut, kutsutaan **nextStaten onEnter**-metodia. Jos tila ei ole vaihtunut, tilakone vain kutsuu **onUpdate**-metodia kunnes tila vaihtuu.

1. **waitingState**

Odottaa käyttäjältä joko hiiren tai näppäimistön painalluksia. Pelin saa pauselle painamalla P-näppäintä, josta peli siirtyy tilaan **pausedState**. Vastaavasti kun hiirtä painetaan, siirtyy tilakone tilaan **loadingThrowState**. Pelin lähtötila on **waitingState**. **waitingState**lla on kaksi eri **onEnter**-metodia, toinen ensimmäistä **onEnter**-tilaa varten ja toinen tilanteita varten jotka tulevat pelin aikana. Esimerkiksi **onEnter** päävalikosta antaa **waitingState**lle parametriksi tason, jota pelaaja pelaa, minkä perusteella **waitingState** lataa .csv-tiedostosta tiedot pelattavasta tasosta. Vastaavasti **onEnter** kesken pelin eli ilman tasoa parametrina vain päivittää peliä. Aikakatkaisusta tai esc-näppäimestä siirrytään suoraan **gameEndedState**en.

2. **loadingThrowState**

Pelissä tämä tarkoittaa tilaa, jossa hiiri on painettu pohjaan ja pelin ritsaa venytetään taaksepäin, kunnes hiiri vapautetaan ja lintu lähtee ritsasta. Ohjelmassa tämä tarkoittaa sitä, että peli on **loadingThrowState**ssa, kunnes hiiri vapautetaan. Ennen hiiren vapauttamista päivitetään ritsassa olevan linnun paikkaa, jonka avulla määritetään linnun tuleva nopeus ja suunta. Kun hiiri lopulta vapautetaan, siirtyy **loadingThrowState** **onExit**iin, jossa pelin fysiikkamoottorille annetaan uusi objekti, jolla on nopeus, paikka, suunta ja linnun tyyppi.

3. **onAirState**

Tila, jossa lintu on jo heitetty, eikä pelaaja voi tehdä mitään muuta kuin mahdollisia lintukohtaisia lisäominaisuuksia (esim. linnun nopeuden muutos tms.). Tässä tilassa päivitetään pelin tilannetta, eli esimerkiksi törmäyksiä ja mahdollisia objektien tuhoutumisia. Kun kaikki kartalla olevat objektit ovat

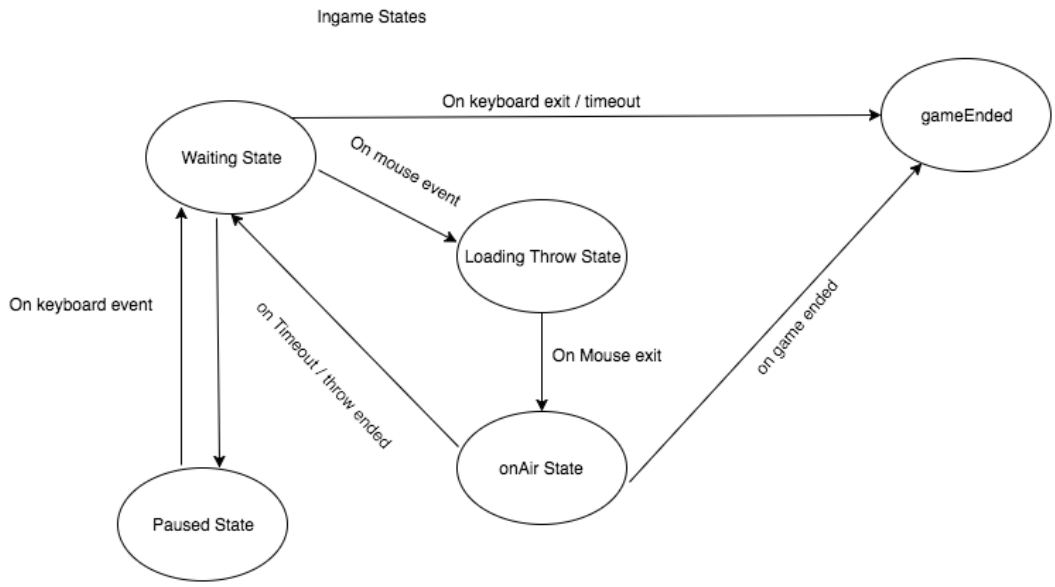
pysähtyneet tai aikaraja täyttyy, menee tila **onExit**iin, jossa se tarkastelee onko peli päättynyt (kaikki objektit tuhottu tai linnut loppuneet kesken). Jos peli on loppunut seuraava tila on **gameEndedState**, jos taas lintuja on jäljellä ja kaikkia objekteja ei vielä tuhottu, siirtyy peli **waitingStateen** odottamaan uutta heittoa. Ennen seuraavaan tilaan siirtymistä kutsutaan **Points-**luokasta pisteiden päivitystä.

4. **pausedState**

Tila johon päästään painamalla P-näppäintä. Tilassa pysäytetään kaikki, kunnes pelaaja painaa uudestaan näppäintä P ja peli jatkuu.

5. **gameEnded state**

Tila, johon päädytään kun peli on loppunut, tilasta siirrytään takaisin päävalikkoon. Tila antaa yhteenvedon pelin kulusta (pisteet, taso suoritettu yms.).



3.3. Fysiikkamoottori

Fysiikkamoottorin pelille rakennamme itse. Näin pidämme huolen siitä, että kaikki tarvitsemamme ominaisuudet ovat helposti käytettävissä ja muokattavissa. Käytämme yksinkertaisia muotoja, kuten palloja, nelikulmioita ja kolmioita törmäyksen tunnistamiseen ja painovoimakäyttäytymisiin, jolloin saamme pidettyä fysiikkamallinnuksen realistisen yksinkertaisena tämän projektin aikataululle. Tärkeimmät ominaisuudet fysiikan kannalta ovat painovoima, sekä vektorilaskenta heitoille. Haastavampi osuus tulee muodostumaan törmäyksen tunnistamisesta ja sen vaikutuksista, mutta nämäkin seuraavat fysiikan lakeja ja verrattain yksinkertaista vektorilaskentaa. Tarkoitus on saada projektin alkuvaiheessa luotua moottori joka ei vielä anna visuaalista kuvaa, vaan objektien liikettä tarkkaillaan koordinaatiston avulla. Kun tämän on tarpeeksi toimivassa kunnossa ruvetaan implementoimaan graafikan piirto-ominaisuuksia fysiikkamoottorin päälle.

3.4. Grafiikka ja muu media

Graafinen ilme pelille tulee olemaan piirettymäinen ja käytämme kaikkien graafisten kappaleiden suunnittelussa yksinkertaisia muotoja, fysiikkamoottoria mukaillen. Käytämme vektorigrafiikka aina mahdollisuuksien mukaan, jolloin peliä on helpompi skaalata jokaiselle näytölle. Käytämme siis mielikuviittelisena koordinaatistona joko aluetta -1...1 tai 0...1. Grafiikan piirto saa kaikki tarpeelliset parametrit objektien sijainnista ja asennosta suoraan fysiikkamoottorilta, jolloin ohjelman tämä osio ei suorita varsinaisia laskutoimituksia lähes ollenkaan. Käytämme grafiikan piirtämiseen SFML-kirjastoa sen monipuolisuuden ja helppokäyttöisyyden vuoksi. Samalla kirjastolla saamme myös äänet toimimaan pelissä, mutta äänimaailman tarkempi suunnittelu jää projektin myöhemmille viikoille. Käyttöliittymän osalta valikot tulevat sisältämään hyvin yksinkertaisia valintoja kuten esimerkiksi ääniasetukset ja kentänvalintaikkunan. Itse pelissä näkyviä käyttöliittymäelementtejä ovat hiiren kursori sekä pistelaskenta.

3.5 Pelin kulku

Pelin idea on varsin yksinkertainen. Pelaaja saa x määrän eri tyyppisiä lintuja käytettäväkseen per kenttä. Pelaaja sen jälkeen ampuu linnut ritsalla kohti maalialuetta, joka sisältää eriarvoisia maaleja (sikoja alkuperäisessä Angry Birdissä), sekä ominaisuuksiltaan erilaisia esteitä. Nämä esteet voivat erota toisistaan esimerkiksi kestävyydeltään, kitkaltaan tai massaltaan. Jokainen maali ja este antavat pisteitä sen perusteella kuinka paljon niihin tehtiin vahinkoa, mutta kentän voittaa vain tuhoammalla kaikki maalit. Pisteitä tallennettaan globaaleihin muuttujiin ja ne näkyvät pelaajalle koko ajan. Kun kaikki maalit on tuhottu, saa pelaaja vielä pisteitä käyttämättömistä linnuista, jonka jälkeen peli loppuu ja siirtyy pisteruudun kautta takaisin käyttäjän valinnasta joko päävalikkoon tai seuraavalle tasolle.