Juho Sillanpää

589903

Tornipuollustus

Tekninen suunnitelma

Tekninen suunnitelma

Ohjelman rakennesuunnitelma

Luokkajako:

- Game
  - peliolio, kasaa kokonaisuuden yhteen
- Filereader
  - lukee pelitiedostosta tiedot pelin karttakuvasta (method: read_map)
  - lukee tiedostosta tiedot kierrosten vihollistyypeistä ja niiden määristä per kierros
- Map
  - kartta joka koostuu pienistä ruuduista
  - kuvaa pelikentän karttakuvaa, luodaan read_map- etodin pohjalta
  - tietää onko tietty ruutu vapaa, onko siinä torni, kuuluuko tiehen ja onko tietyn tornin kantaman sisällä (hoituu distance luokan avulla)
- Round
  - kuvaa kierroksella tulevien vihollisten määrät ja nopeutta jolla ne lähetetään
  - lähettää kierroksen vihollisoliot liikenteeseen
- Tower:
  - alaluokkina eri tyyppisten tornien luokat
  - ampuu vihollisia
  - pitää kirjaa tuhoamistaan vihollisista
  - tornilla tietty ampumisnopeus ja tuhovoima- Näitä voidaan kehittää luokassa olevien metodien avulla
- Enemy
  - kuvaa vihollisyksikköä, sillä on tyyppi, reitti, nopeus sekä elämä(pisteet)
  - Liikkuu reitillä vakionopeudella
  - Kun olion elämät loppuvat niin olio tuhoutuu
- Window
  - luo graafisen ikkunan ja päivittää sitä kokoajan pelin edetessä
  - Päivittyy kokoajan kun pelialueen kartta muuttuu
- GraphicsItem
  - Luo torneja, vihollisia ja kartan ruuduille grafiikat
- Route
  - Kuvaa tietyn vihollisolion reittiä, reittejä useampi, joten kaikki eivät kulje samaa reittiä
  - reitin käännökset pyöristettyjä, eli meno ei ole kulmikasta
- Distance

- Apuluokka jonka avulla tornin Map-luokka merkitsee tietyn tornin kantaman sisällä olevat ruudut

Vaihtoehtoinen ratkaisu sille miten tietyn kierroksen vihollisjoukot luodaan olisi helpohko kierroksen numeroon (monesko kierros) perustuva algoritmi, mutta tekstitiedoston avulla saadaan tehokkaammin luotua kierroksia jotka poikkeavat täysin muista kierroksista. Näitä voitaisiin luoda if kierros = x-rakenteiden avulla, mutta se olisi hiukan jäykempi tapa. Tekstitiedoston avulla saadaan myös suunniteltua tietylle tasolle (eli tietylle kartalle) täysin omanlaisia kierroksia. Kolmas syy oli, että kierrokseen perustuva menetelmä olisi ollut melko alkeellinen.

Koska graafisen ikkunan tulee päivittyä kaiken aikaa, voi olla, että tätä varten tulee luoda oma luokkansa. Tämä vaatii hieman lisää perehtymistä siihen, miten kyseinen ominaisuus saadaan toimimaan. Sama pätee siihen, miten saadaan eri toiminnot (vihollisten liike) riippumaan ajasta.

Käyttötapauskuvaus

Kun pelaaja käynnistää pelin niin main-ohjelma luo graafisen valikon, josta pelaaja valitsee haluamansa tason (ei vaikeustaso). Tämän jälkeen luodaan Game- luokan olio, joka kuvaa kyseistä peliä. Seuraavaksi kyseistä tasoa vastaava tekstitiedosto avataan Filereader luokan metodien avulla ja Map-luokka luo sen pohjalta tasoa vastaavan graafisen karttakuvan ja liittää sen Game-olioon. Pelin alustuksen yhteydessä myös eri kierrosten vihollisten tyypit ja määrät alustetaan Round-luokassa. Jokaista pelin kierrosta kohti luodaan oma olio, joka sisältää tiedot eri vihollisista, niiden määristä sekä siitä mitä vauhtia niitä lähetetään reitille. Nämä tiedot luetaan myös Filereader- luokassa.

Viholliset ovat Enemy-luokan olioita, jolla on jokin tiettytyyppi sekä elämä ja nopeus. Eri tyyppejä varten näillä on myöskin omat luokkansa, joista haetaan esimerkiksi niiden elinvoima ja nopeus. Näiden tyyppien pohjalta myös luodaan, kaikille enemy-olioille omat graphicsItemit siihen erikoistuneessa luokassa.

Tämän jälkeen Window-luokka luo ikkunan, jossa näkyy pelialueen karttakuva ja muita esisuunnitelmassa näkyviä tietoalueita. Alustuksen yhteydessä pelaajalle (eli luodulle game-oliolle) alustetaan rahaa x määrä. Tämän avulla pelaaja voi rakentaa tornin, jollekin kartan paikalle. Torni on Tower-luokan olio, jolla on tyyppi (tietty ampumisnopeus ja tuhovoima) ja sijainti kartalla. Kun torni-olio asetetaan peliin, niin game-olion kartasta varataan sille se paikka, johon torni asetetaan. Tornin hinnan verran rahaa tietenkin poistuu game-olion rahapussista.

Lisäksi kun torni olio asetetaan kartalle, niin Distance-luokan metodi aktivoi kartalta alueen, joka on kyseisen tornin kantaman sisällä tornista. Kun vihollisolio saapuu tälle alueelle, niin torni saa tiedon siitä ja asettaa kyseisen olion kohteekseensa.

Pelaajan painaessa ”start next round” pushbuttonia Round-luokan metodi send_troops lähettää kierroksen vihollisoliot matkaan. Vihollisolioiden reitti luodaan Route-luokassa. Eri oliolla on hieman eri reitit kartalla, eli ne kulkevat hieman eri kohdassa ”tietä”. Jokaisella vihollisoliolla on siis myös reitti ja se tietää oman sijaintinsa kartalla.

Torni Tornin havaitessa vihollisen sen kantaman sisällä se alkaa ampua sitä, jolloin enemy-olion elämä(hitpoints) alkaa pienentyä. Kun hitpoints on nolla tai pienempi, niin enemy olio tuhoutaan Enemy-luokan destroy-metodilla ja torni-olio valitsee seuraavan kohteensa. Mikäli enemy-olio selviää radan loppuun asti tukikohtaan asti, niin itse pelaajan elinpisteet vähentyvät. Kierroksen kaikkien vihollisten

tuhouduttua tai päästyä läpi, game-olion rahamäärä lisääntyy jollakin summalla ja ”start next round”-pushbutton ilmestyy uudelleen. Game-olio pitää koko ajan kirjaa siitä, mikä kierros on menossa, ja tässä vaiheessa kierros kasvaa yhdellä.

Algoritmit

Distance- luokka käyttää hyväkseen pythagoraan lausetta selvittääkseen kahden ruudun välisen etäisyyden.

Vihollisten reitti taas määräytyy tekstitiedostossa jo ja ja Route-luokka jakaa sen periaatteessa vain suoriksi viivoiksi ja viivojen risteyskohdat pyöristetään.

Vihollisolioiden liikuttamiseen tarvittavat algoritmit ja graafisen käyttöliittymän päivitykseen liittyvät toiminnot eivät ole vielä selvillä. Aihe on tarkoituksellisesti valittu sellaiseksi, joka kiinnostaa, mutta johon liittyen täytyy itsenäisesti opiskella asiaa.

Tietorakenteet

Pelialueen kartta muodostuu listasta, jotka sisältävät listoja ja jokaista listan alkiota vastaa jokin graphicItem.

Jokaisella yksittäisellä ruudulla on lista, jossa on alkioina ne torni-oliot, joiden kantaman sisällä se on.

Aikataulu

1. viikko
    a. ensisilmäys siihen miten saadaan aika mukaan ohjelman toimintaan ja miten graafinen käyttöliittymä saadaan toimimaan sekä tutustuminen pyQt kirjaston erilaisiin ominaisuuksiin
    b. luokkien luominen ja niihin init metodit + tärkeimmät metodit (ei sisältöä)
    c. 1. tekstitiedoston kirjoittaminen
2. viikko
    a. readerin, Enemyn ja mapin koodaaminen
    b. towerin ja sen alaluokkien koodaus
    c. mainin hahmottelu
3. viikko
    a. Game-luokan koodaus sekä route ja distance
    b. lisää graafisen käyttöliittymän opiskelua
4. viikko
    a. window ja graphicsitem
5. viikko
    a. graphicsitem
6. viikko
    a. koko paketin testausta kokonaisuutena ja muutosten tekemistä, mikäli osat eivät toimi halutulla tavalla
7. viikko
    a. koko paketin testausta kokonaisuutena ja muutosten tekemistä, mikäli osat eivät toimi halutulla tavalla
8. DL!!

Kaikilla viikoilla isossa osassa

Yksikkötestaussuunnitelma

Kun jokin kokonaisuus on tehty, luodaan testausmetodi, joka testaa koodin toimimista. Esimerkiksi kun luetaan tiedostoa, niin lukemisen jälkeen tarkistetaan, onko read-metodit luoneet oikeat vihollisoliot ja onko esimerkiksi kartan tie siellä missä sen kuuluisikin olla. Tornin kantama saadaan selville helposti lisäämällä koodiin metodi joka värjää kantaman sisällä olevan alueen esimerkiksi punaiseksi, jolloin koodia ajamalla näkee heti, mikäli tämä osa ohjelmasta menee pahasti pieleen. Koska ohjelma toimii graafisen käyttöliittymän avulla, niin moni asia saadaan testattua helposti ajamalla itse ohjelmaa.

Kirjallisuusviitteet ja linkit

Aion hyödyntää paljon youtube-videoita uusien asioiden opettelemiseen. Youtubessa on todella paljon kanavia, jotka ovat erikoistuneet koodaamisen opettamiseen.

Lisäksi hyödyllisiä sivustoja:

Stackoverflow on todella hyödyllinen pienempien ongelmien ratkaisussa

http://programarcadegames.com/index.php?chapter=introduction_to_graphics

https://www.tutorialspoint.com

https://wiki.python.org

https://docs.python.org