Ella Hakala 473527

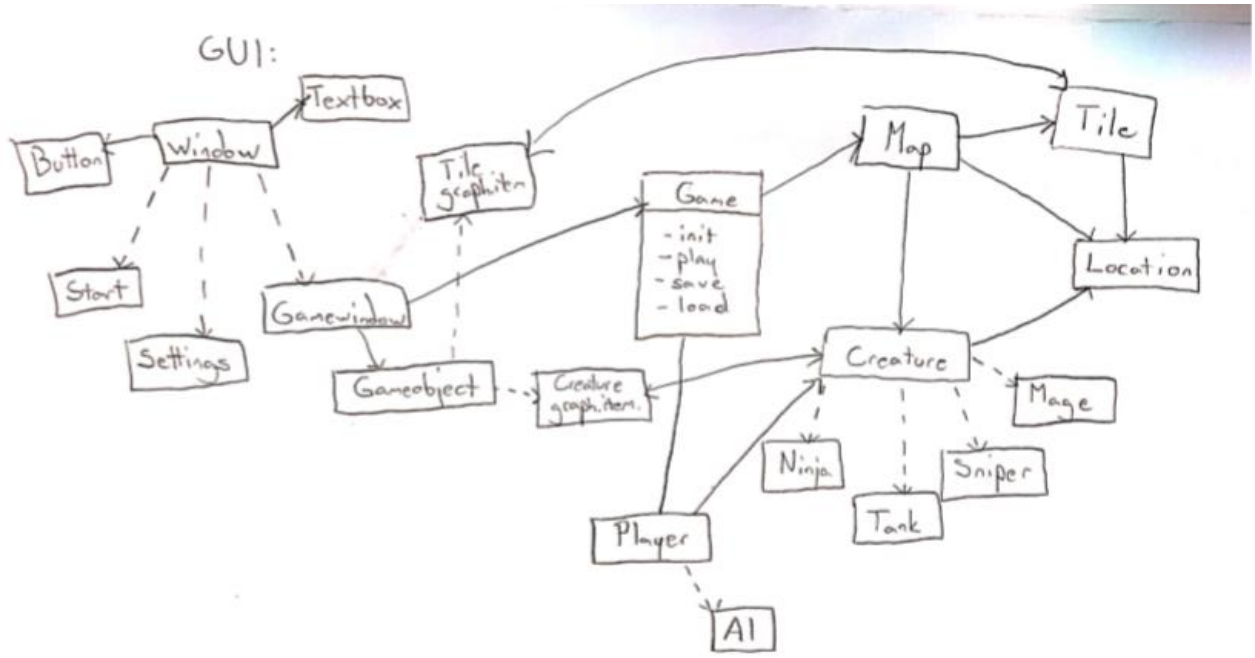
BioIT/ ELEC, 4. vsk

05.03.2019

Python Y2 projektin tekninen suunnitelma

Strategiapeli

- **Ohjelman rakennesuunnitelma**



Yllä karkea suunnitelma luokkarakenteista. Kaavio ei välttämättä noudata täysin uml-protokollaa, esim nuolten suunnat eivät välttämättä ole oikein päin.

Graafinen käyttöliittymä koostuu luokasta Window, jolla on alaluokat Start, Settings ja Gamewindow, jotka vastaavat pelin kolmea eri näkymää. Ikkunoilla on ainakin attribuutteina luokat Button ja Textbox, joiden avulla ne saadaan toimimaan halutulla tavalla. Gamewindow:lla on myös attribuuttina lista Gameobjecteja, jotka ovat grafiikkaobjekteja, näitä ovat alaluokat Creature_graphicsitem ja Tile_graphicsitem vastaamaan liikuteltavia hahmoja pelikentällä, sekä pelikentän tiiliä, jotka voivat olla erinäköisiä riippuen tiilen tyypistä.

Koko pelattavaa osuutta kuvaa luokka Game, jolla on mm. metodit init, play, save ja load, pelin alustamista, pelaamista, tallentamista ja lataamista varten. Game sisältää pelikentän Map, joka koostuu tiilistä Tile. Kaikkien objektien lokaatiota kartalla kuvataan luokan Location avulla. Peli sisältää myös pelaajaluokan Player, johon on tallennettu mm. pelaajan vuoroon ja tiimiin liittyvät tiedot. Pelaajaluokalla on alaluokka AI, joka suorittaa vuoron toiminnot itsekseen eikä ota niitä inputteina käyttäjältä. AI-luokkaan on sisällytetty myös

Ella Hakala 473527

BioIT/ ELEC, 4. vsk

05.03.2019

”päätöksentekoon” vaikuttavat metodit, joilla lasketaan mihin tekoälypelaaja liikuttaa hahmojaan ja miten ne hyökkäävät. Sekä pelaajilla että kartalla on tieto pelissä mukana olevista hahmoista, joita kuvaa luokka Creature. Creaturella on alaluokkina eri hahmotyypit.

- **Käyttötapauskuvaus**

Pelaajan on tarkoitus pelin käynnistäessä nähdä aloitusikkuna, jossa on painikkeina vaihtoehdot Play, Settings ja Quit. Pelaaja valitsee suoraan vaihtoehdon Play, eli uusi peli käynnistetään oletusasetuksilla. Avautuu peli-ikkuna, jossa ihmispelaajan pelinappulat ovat ruudun oikealla puolella, ja tietokonevastustajan nappulat vasemmalla. Teksti-ikkuna kertoo että pelaaja voi aloittaa vuoronsa, jonka jälkeen ikkunassa kerrotaan mitä nappulaa pelaajan on tarkoitus liikuttaa. Mahdolliset ruudut joihin nappulaa voi liikuttaa näytetään värillisenä, ja pelaaja voi liikuttaa nappulaa klikkaamalla jotakin ruutua. Liikkuminen on suoritettu ja on aika liikuttaa seuraavaa nappulaa, joka kerrotaan jälleen teksti-ikkunassa. Sama toisstuu kunnes pelaajan kaikki nappulat on liikutettu. Tämän jälkeen teksti-ikkunassa pyydetään pelaajaa valitsemaan hyökkäävä nappula klikkaamalla. Klikatessa jotakin nappulaa ilmestyy valikko kyseisen hahmotyypin mahdollisista iskuista, joista pelaaja voi valita yhden. Kun hyökkäys on tehty, sen vaikutukset suoritetaan (vahinkoa vastustajille, status-elementit tms.) ja vuoro siirtyy tietokonepelaajalle (kerrottu teksti-ikkunassa). Tietokonepelaaja suorittaa samat toiminnot kuin ihmispelaaja, mutta itsekseen. Näin jatketaan, kunnes jommankumman pelaajan kaikki nappulat on tuhottu. Tällöin tulee näkyviin joko voitto- tai häviöteksti ja siirrytään takaisin päävalikkoon.

- **Algoritmit**

Koska kartta on ruudukkopohjainen ja pelaajien liikkuminen voi tapahtua vain x- tai y-suunnassa (ei vinottain), etäisyyksien laskeminen on yksinkertaista. Ongelmia tulee kuitenkin jos tiellä on esteitä, joten pohdin että tekoälyn reitinvalinnan voisi laskea esimerkiksi BFS-algoritmin avulla (breadth-first search). Reitin valintaan implemetoidaan myös tiettyjä prioriteettejä, esimerkiksi heikommat pelaajat pyritään siirtämään turvaan esteiden taakse.

- **Tietorakenteet**

Staattiset taulukot toimivat hyvin pelin datan käsittelyyn, sillä maksimikoot (kartan koko, pelaajien määrä) on määritelty ennalta joten niiden ei tarvitse olla dynaamisia. Pelitilanne tallennetaan Python MySQL:n avulla tietokantaan, joka tehdään itse. Tietokannan tulee sisältää tieto kartasta, kartalla olevista nappuloista, nappuloiden tilanteet (hp), ja kumman vuoro on seuraavaksi (tallentaa ei voi kesken vuoron).

- **Aikataulu**

Ensimmäinen prioriteetti on saada peli-ikkuna toimimaan niin, että kartta toimii niin kuin pitääkin ja hahmoja voi liikutella klikkaamalla kartalla.

Sen jälkeen alan työstämään pelin sisäisiä toimintoja eli hyökkäyksiä, vuoropohjaista liikkumista sekä AI:n toimintaa. Kun itse pelikokemus on valmis voi lähteä työstämään

pienemmän prioriteetin toimintoja eli käyttäjäystävällisyyden parantelua. Silloin implementoin esimerkiksi tallennuksen, kartan kustomoinnin ja valikot.

- **Yksikkötestaussuunnitelma**

Koko ohjelman testaus suoritaan edellä kuvatussa järjestyksessä. Aina jokaisessa vaiheessa on tärkeää saada implementoidut ominaisuudet toimimaan kunnolla, ennen eteenpäin jatkamista. Testaus toimii lähinnä pelin käyttämisellä mahdollisimman monilla eri tavoilla. Pelin rakenne ei onneksi anna pelaajalle hirveästi vapauksia toimia virheellisesti, mutta esimerkiksi virheellisten tiedostojen syöttämistä tulee testata. AI:n toiminta pitää myös suunnitella tarkasti, ettei tietokonepelaaja voi yrittää siirtää nappuloita vaikkapa ruudukon ulkopuolelle tms.

- **Kirjallisuusviitteet ja linkit**

Toistaiseksi olen tutustunut pygameen tästä tutoriaalista:

https://realpython.com/pygame-a-primer/

ja pythonin kanssa tietokantojen käyttöön ja luomiseen tästä linkistä:

https://www.w3schools.com/python/python_mysql_create_db.asp