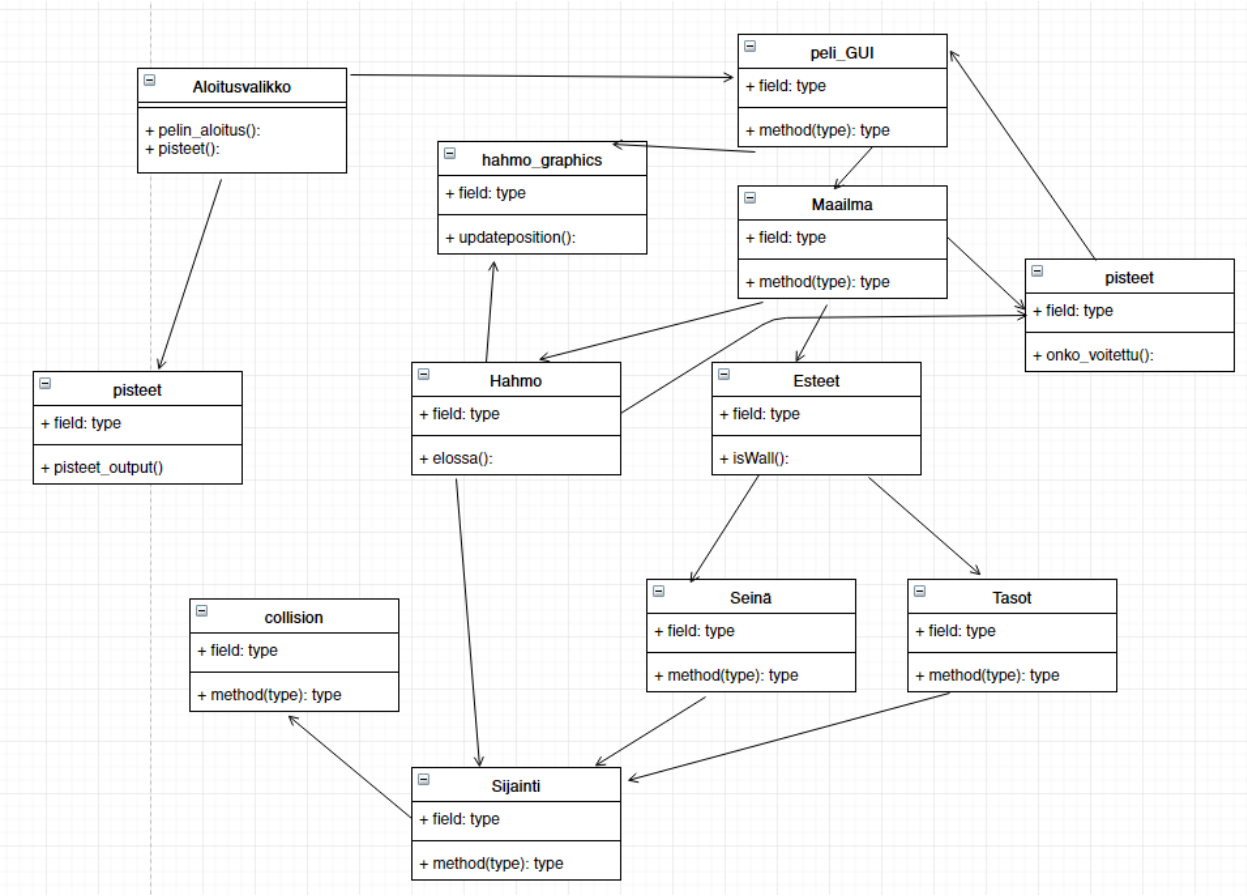
Tekninen suunnitelma

Rakennesuunnitelma



Valikko:

Luo ja piirtää aloitusvalikon. Sisältää esim. painikkeen pelin aloittamiselle.

Pisteet:

Sisältää pisteiden tallennuksen ja niiden tarkastelun.

GUI:

Luokka peli_GUI piirtää pelin näytölle. Metodi update_hahmo päivittää hahmon sijainnin näytölle.

Maailma:

Sisältää pelimoottorin, joka luo tason ja päivittää peliä suhteessa aikaan. Lähettää tietoa GUI:lle. Pelimoottori toteutetaan looppina, joka keskeytetään, jos voittamisen ehto täyttyy.

Hahmo:

Sisältää hahmon liikkumiseen liittyvät metodit. Lisäksi luo hahmon.

Sijainti:

Sisältää sijaintitietoa. Tätä luokkaa hyödyntävät esteet ja hahmo.

Seinä:

Perii este-luokan. Sisältää seinät, lattian ja katon pelille.

Tasot:

Perii este-luokan. Sisältää tasot, jonka päällä hahmo voi hyppiä.

Pisteet:

Sisältää pisteiden laskun. Tämä toteutetaan viimeiseksi, koska ei voittamisen sisälly vaatimuksiin.

Käyttötapakuvaus

Käyttäjä avaa ensiksi aloitusvalikon, jonka aloitusvalikko-luokka hoitaa. Käyttäjä aloittaa pelin painamalla aloitusnappulaa.

Tämän jälkeen piirretään maailma ja alkusijainti hahmolle. Tämä tapahtuu maailma-luokassa, jossa käsitellään esteiden ja pelaajan sijainnit. Esteiden sijainti pysyy samana – ainoastaan pelaajan hahmo liikkuu kentällä. Maailma-luokka hoitaa myös loopin-jossa peli pyörii.

Pelaaja liikuttaa hahmoaan nuolinäppäimillä ja sen sijaintia muutetaan sijainti-luokassa. Tarkistetaan myös että hahmo ei liiku esteiden läpi. Hahmon sijainti päivitetään näytölle.

Pelaaja liikuttaa hahmoaan maaliin, jossa peli päättyy tai jos halutaan lisätä useampia maailmoja, niin vaihdetaan maailmaa. Näytölle päivitetään peli_GUI-luokan avulla. Voittamisen yhteydessä piirretään voittoilmoitus ja kysytään pelaajanimeä pisteiden tallennukseen.

Tietorakenteet

Omasta mielestäni listojen käyttö on riittävää toteutuksessani, sillä esimerkiksi esine-objektien iterointiin se riittää hyvin.

Sijainti kuvataan kaksiuloitteisella listalla, koska sen arvoja muutetaan ja tuple-rakenteella se ei onnistu.

Monimutkaisempia rakenteita ei mielestäni tarvita.

Aikataulu

Aloitusvalikon tekemiseen ja testaamiseen menisi pari tuntia, koska se ei ole kovin monimutkainen; luodaan vain ikkuna, johon tulee pari nappulaa.

Tämän jälkeen tehdään maailma ja gui. Luodaan ensimmäinen kenttä. Tämä sen takia, että siihen päälle on kätevämpää muodostaa hahmoon liittyvät ominaisuudet. Samassa luodaan este-luokat, sekä sijainti-luokka. Tähän arvioisin menevän ~10 tuntia.

Luodaan hahmo ja sen piirtäminen ~2 tuntia.

Pelin päivittäminen ja liikkuminen ~10 tuntia, koska luodaan collision detection ja ns. pelimoottori, joka hoitaa pelin reaaliaikaisen päivittämisen.

Pisteiden laskeminen ja voittoehdon tekeminen. Tähän noin pari tuntia.

Pisteiden tarkastelu ja tallentaminen ~5 tuntia.

Yksikkötestaussuunnitelma

Kun maailma luodaan, niin testataan, että ohjelma piirtää esteet oikealle sijainnille ja lisäksi, ettei se salli esteiden päällekkäisyyttä. Esimerkiksi hahmoa liikuttamalla pitkin tasoa, voidaan helposti huomata törmäysten hoitavan metodin toimintaa.

Linkit ja viitteet

pyqt.sourceforge.net/Docs/PyQt5/ - PyQt5 dokumentaatio

https://plus.cs.hut.fi/y2/2018/ - kurssin materiaali

https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects - collision detection