Ohjelmoinnin peruskurssi Y2 CS-A1121

1) Tasohyppelypelin projektisuunnitelma: 24.2.2021

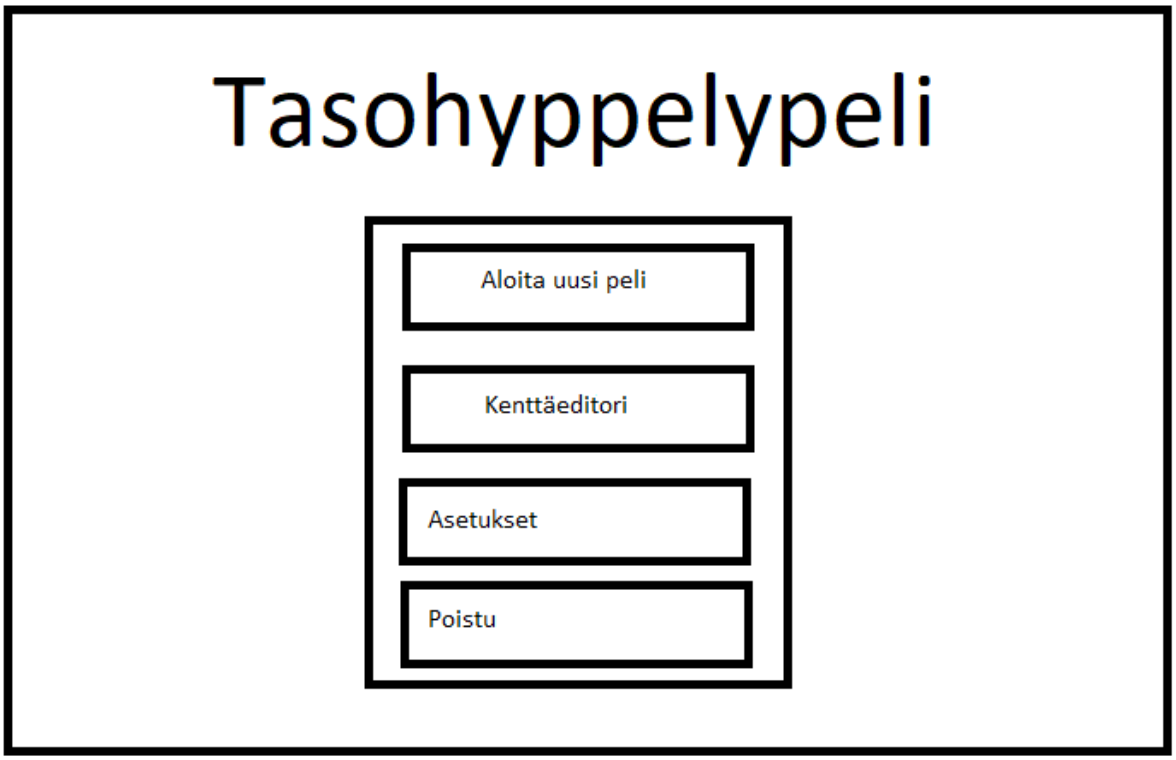
Tekijä: Aaro Reijola, 731065 (AUT 2019)

2) Yleiskuvaus ja vaikeustaso

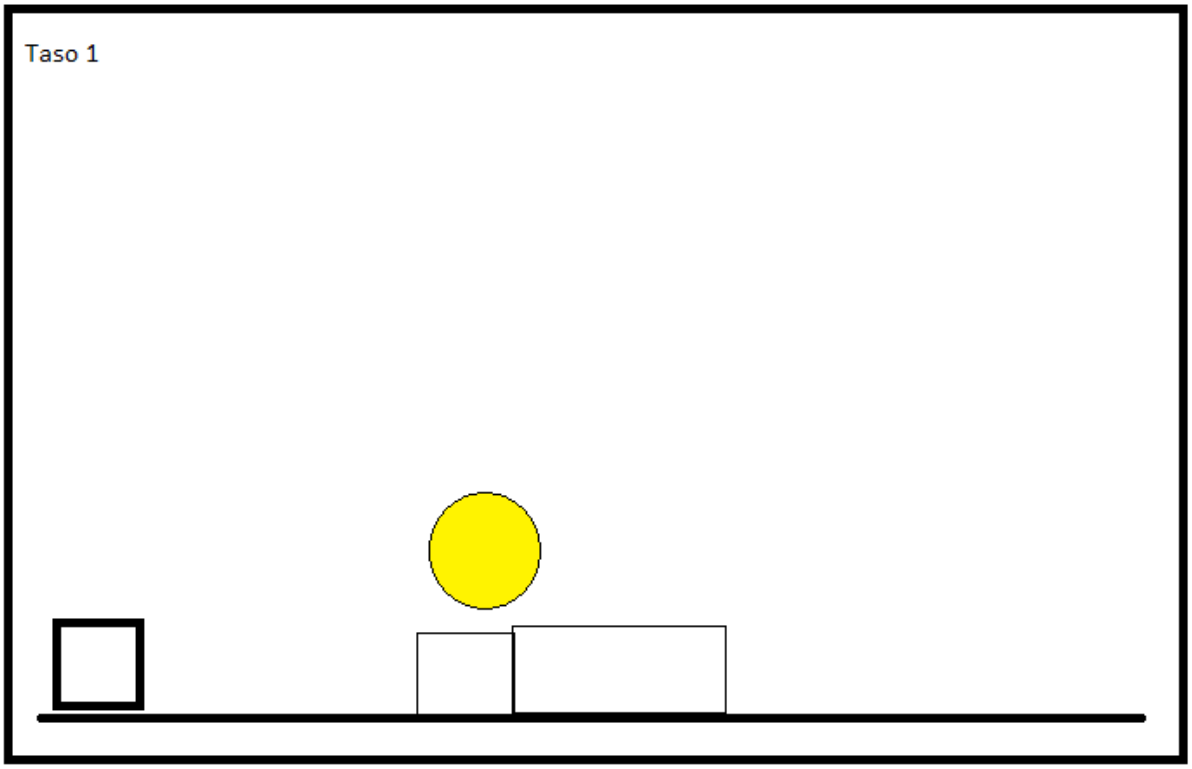
Projektin tavoite on luoda tehtävänannon 9.2 mukainen tasohyppelypeli, joka täyttää vaativat vaatimukset. Tasohyppelypelissä pelaajan tavoite on yksinkertaistettuna kerätä kaikki kolikot ja selvitä maaliin asti kuolematta. Hahmon kuollessa taso alkaa pelaajan niin halutessa alusta.

3) Käyttötapauskuvaus ja käyttöliittymän luonnos

Ohjelma sisältää yhden ikkunan, jonka keskiössä on pelin graafinen osa. Ohjelma saa valikossa syötteensä joko nuolinäppäimillä, tai hiirellä ja pelin aikana nuolinäppäimillä. Neliön muotoista pelihahmoa ohjataan WASD-näppäimillä, jossa W saa hahmon hyppäämään ja S kyyristymään. Ohjelman pelaajalle välittämä informaatio on pitkälti valikoiden nappien tekstejä, jotka eivät isompia selittelyjä kaipaa.



Kuva 1.1: Pelin päävalikko (pelkistetty)



Kuva 1.2: Pelin aikainen tilanne näytöllä

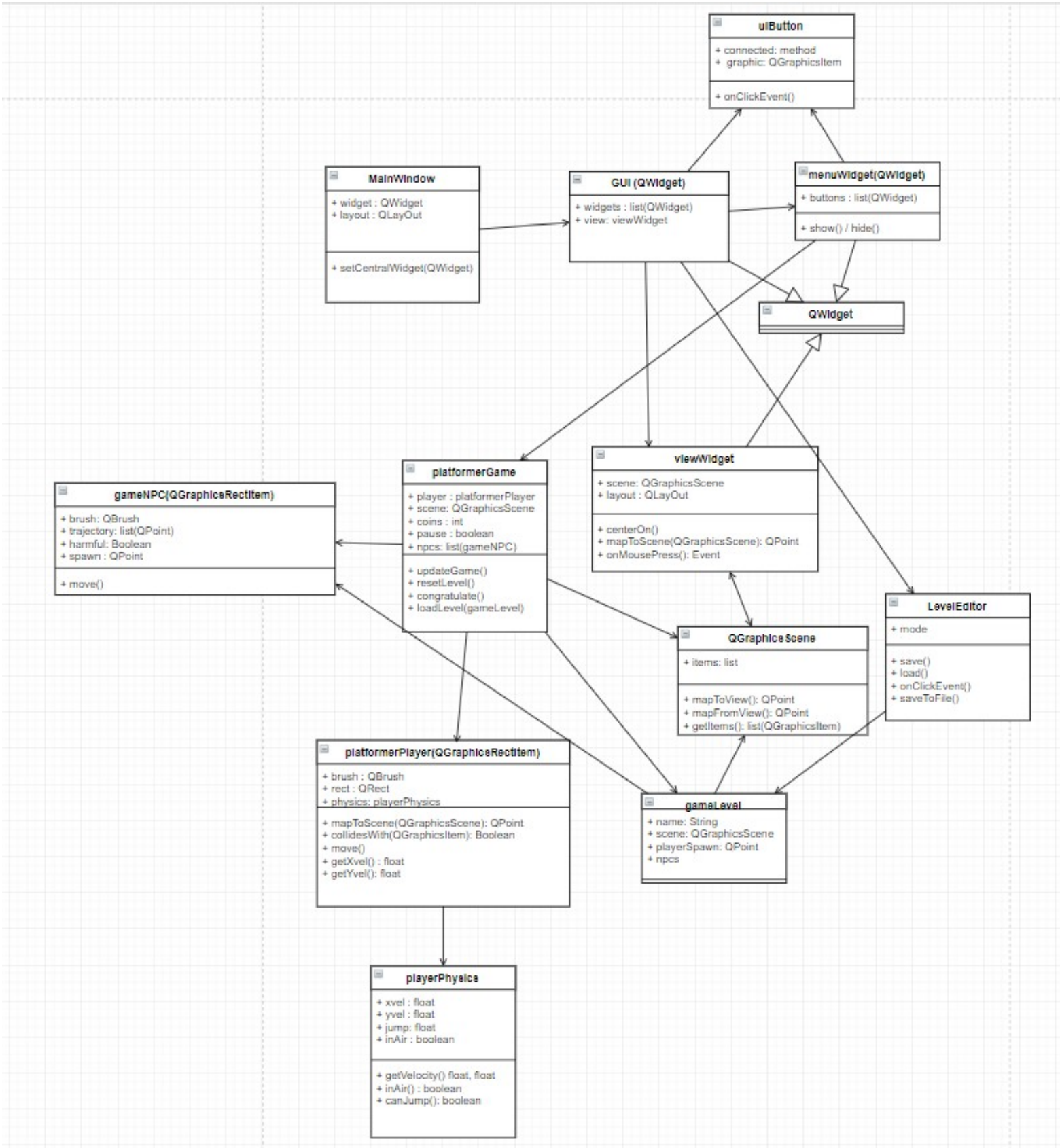
Käyttäjän käynnistäessä ohjelman ensimmäisenä avautuu päävalikko, josta voidaan valita kenttäeditorin, asetusten tai pelaamisen väliltä. Valikko toimii laatikoita klikkaamalla tai sitten valitsemalla WASD- ja ENTER- näppäimillä haluttu vaihtoehto. Kun aloitetaan peli, näyttöön piirtyy ensimmäinen taso (tutorial), jossa pelaajalle ohjeistetaan pelin mekaniikat, tavoitteet sekä ohjaus. Esc-näppäimestä saadaan pelin aikanakin avattua valikko, josta voidaan muuttaa asetuksia sekä sulkea koko peli. Valikon avaaminen pausettaa pelin. Ensimmäisen tason jälkeen pelaaja ohjataan joko pois sovelluksesta tai luomaan omia tasoja.

4. Ohjelman rakennesuunnitelma

Ohjelma sisältää MainWindow-luokan, jonka CentralWidgetin lapsiksi eri näytöllä näkyvät elementit lisätään. GUI sisältää pelin aikana view:n, joka toimii ikkunana pelin sceneen, jossa pelihahmo ja tason eri osat elävät. Scene on linkki pelilogiikan ja pelin grafiikkojen välillä. Scenen sisältämien grafiikkaelementtien avulla voidaan selvittää loppuuko peli, voittaako pelaaja ja vaikkapa osuuko pelaaja piikkeihin. PlatformerGame-luokka hoitaa pelin liikkuvien osien liikuttelun aina kun update()-metodia kutsutaan ajastetusti.

Pelin ulkopuolella GUI sisältää menun, joka toteutetaan menuWidget-luokalla ja sen sisältämillä napeilla, jotka liitetään haluttuihin toimintoihin.

LevelEditorissa hoidetaan tasojen tallentaminen tiedostoihin sekä niiden lukeminen ja lataaminen pelattavaksi. Jokainen taso sisältää tiedot tason sisältämistä grafiikkaobjekteista, non player charactereista (NPC), pelaajan aloituskoordinaateista, kerättävien kolikoiden määrästä ja sijainnista sekä maalista.



Kuva 1.3: UML-kaavio ohjelman käyttämistä luokista

5. Tietorakenteet

Pythonin ja PyQt5-kirjaston tarjoamat rakenteet riittävät. Pelilogiikan hoitaminen tapahtuu suurimmaksi osaksi seuraamalla qt:n scenellä olevien grafiikkaobjektien välisiä suhteita, joten kirjaston valmiiksi tarjoama lista objekteista on keskeisessä roolissa. Listan käyttö on looginen valinta pelin logiikan joustavuuden kannalta, kun kentän objektien määrä voi vaihdella.

6.Tiedostot ja tiedostoformaatit

Eri tasot tallennetaan erillisiin tekstitiedostoihin, joista data voidaan lukea ohjelman muistiin ja kentästä voidaan tehdä pelattava. Tallennus tapahtuu hakemalla PyQt-kirjaston scenestä kaikki grafiikkaobjektit ja tallentamalla niistä väritiedot, sijainnit, muodot yms tekstitiedostoon. Tämän lisäksi peli saattaa sisältää .png-tiedostoja omien kuvien lisäämiseksi, sekä äänitiedostoja musiikkia ja efektejä varten.

7. Algoritmit

Pelin algoritmit liittyvät hahmojen ohjaukseen ja törmäyksien tunnistuksiin. Ohjelman tehtävä on huolehtia, että pelihahmo liikkuu käyttäjän ohjauksen mukaisesti, ilman tökkivää liikettä tai ”clippausta” kiinteiden esineiden sisälle. Myös objektien koordinaattien täytyy olla tarkasti tiedossa, (esim. mihin kulmaan esineen koordinaatisto kiinnittyy, vai kiinnittyykö keskipisteeseen), jotta törmäyksen tunnistus ja sitä noudattava liike voidaan toteuttaa laadukkaasti. Käytännössä laskennan keskiössä on pythagoraan lause $a^2+b^2=c^2$, jolla pisteiden välisiä etäisyyksiä lasketaan 2D-tasossa. Kaksi esinettä törmäävät, jos niiden grafiikat piirtyvät joltain osin päällekkäin.

Pelin ”monsterit” tai NPC:t noudattavat ennaltamääriteltyä liikerataa, joten ne eivät sisällä mitään varsinaista tekoälyä.

8. Testaussuunnitelma

Ohjelman täytyy kyetä antamaan käyttäjälle kohdan 3 mukainen käyttökokemus. Tätä varten täytyy siis saada aikaan toimiva valikko, GUI, koko ikkuna, sekä pelilogiikka. Ikkunan toimintaa voi testata jo varhaisessa vaiheessa, ja toimivan ikkunan jälkeen siihen voidaan helposti lisätä eri widgettejä, joiden toimivuutta voidaan myös testata asettamalla napeille placeholder-toimintoja testivaiheessa. Tämä mahdollistaa GUI:n osien testaamisen sen mukaan kun ne valmistuvat.

LevelEditorin tallennus- ja lataustoimintoja voidaan myös testata jo ennen koko LevelEditor-toiminnallisuuden valmistumista. Toimiva LevelEditor helpottaa myös lopullisen pelin testaamista, sillä editorilla voidaan luoda helposti erilaisia skenaarioita, joista bugeja voidaan löytää. Lopuksi ennen viimeistä palautuspäivää on tarkoitus antaa parin kaverin pelata peliä tavoitteenaan rikkoa se, jolloin saadaan black-box-testattua ohjelmaa ja mahdollisesti löydetään vielä uusia bugeja korjattavaksi.

9. Kirjastot ja muut työkalut

Koko sovellus tehdään ikkunan ja grafiikan osalta PyQt5-kirjastolla. Kirjasto tarjoaa riittävät mahdollisuudet toteuttaa hahmon ohjaus, collision detection, omien grafiikkaelementtien luominen, koordinaatistomuunnokset ja oikeastaan kaikki mitä tasohyppelypelissä tarvitaan.

Matematiikkaan käytetään math-kirjastoa

Omissa grafiikkaelementeissä käytettäviä pixmappejä luodaan MS Paintilla.

10. Aikataulu

Kurssi on viiden opintopisteen arvoinen, ja 60% arvioinnista koostuu projektista.
1op = 27h, joten projektiin on käytettävissä n. 80h. Aikaa viimeiseen deadlineen on maksimissaan 8 viikkoa, joten tavoite olisi tehdä tasaisesti viikossa kaikkea projektiin liittyvää n. 10H

Ensimmäisen kahden viikon aikana toimiva runko, josta helppo lähteä kehittämään eri asioita osa kerrallaan. Ensimmäisenä ikkuna ja sen sisäiset widgetit, sitten valikot ja muut UI-elementit.
Eri osien testausta suoritetaan sen mukaan kun ne valmistuvat. Ensimmäisenä työpäivänä tavoite saada ohjelma käyntiin ja mahdollisesti kokeiltua muutamaa erilaista widgettiä ikkunassa. Tämän jälkeen omien widgettien luonti ja testaus aikaan saadussa ohjelmassa. Itse peliä testataan viimeisenä, mutta siitäkin tavoite saada runko kasaan jo ennen ensimmäistä checkpointtia, eli aikaa on nollasta noin kolmekymmentä tuntia.

Dokumentaation kirjoitukseen täytyy varata aikaa ainakin 5h. Myös checkpoint ja gitin opettelu vievät aikaa. Itse koodaamiseen jää siis aikaa kokonaisuudessaan ehkä 60h.

Projektin deadlinejä:
-Suunnitelman demo ma 1.3 klo 12

-15.3-26.3 välillä checkpoint sähköpostitse. Tähän mennessä tunteja takana n. 30h

-Demo ennen 21.5

11. Kirjallisuusviitteet ja linkit

Qt:n dokumentaatio: https://doc.qt.io/qt-5/index.html
Pythonin dokumentaatio: https://docs.python.org/3/