Y2 Projekti dokumentti

1. **Henkilötiedot**

   Tasohyppelypeli; Mikko Suhonen – 653871 – EST – 19.4.2019

2. **Yleiskuvaus**

   Tasohyppelypeli, jossa on tavoitteena päästä tason lopussa olevaan portaalin, josta pääsee seuraavaan tasoon. Matkalla on erilaisia esineitä ja vihollisia, joita täytyy varoa. Puolessa välissä tasoa on tallennuspiste ja johon voi palata takaisin niin pitkään kun elämät riittävät. Matkalta löytyy myös kakkuja, joita voi kerätä. Keräämällä 10 kakkua saa uuden elämän. Pelissä on kolme tasoa, joista ensimmäinen on helpoin ja viimeinen vaikein.

   Suunnitelmissa oli vuorikiipeily tyylinen tasohyppely, mutta hylkäsin sen jo alkuvaiheessa, koska en enää tykännyt ideasta.

   Mielestäni ohjelma on 5 arvosanan tasoinen, sillä siinä on toteuttu useita ongelmallisia ominaisuuksia, esimerkiksi kiihtyvä liikkeinen pelaaja ja animaatiot, sekä viholliset, jotka pystyvät reagoimaan pelaajan liikkeisiin.

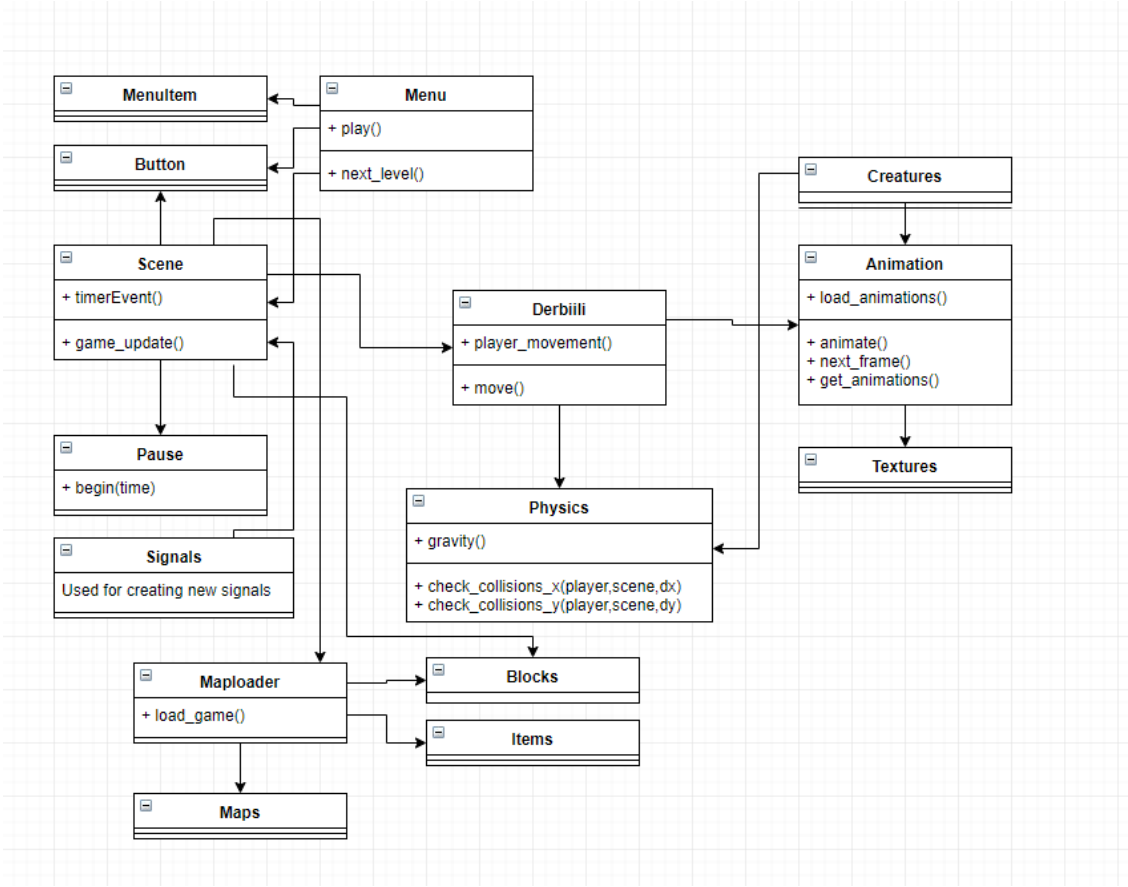
3. **Käyttöohje**

   Ohjelman voi käynnistää suorittamalla main.py tiedoston. Suorittamisen jälkeen olet menu -valikossa, jossa voit käydä vaihtamassa näppäin asetuksia options -valikossa, poistua pelistä tai mennä karttavalikkoon. Kartta valikon kautta pääsee pelaamaan itse peliä valitsemalla halutun tason. Esc -näppäimellä voi palata takaisin menu -valikkoon pelistä.

4. **Ulkoiset kirjastot**

   Ohjelmassa on käytetty ainoastaan PyQt -kirjastoa.

5. **Ohjelman rakenne**



Ohjelman menu valikko pyörii täysin Menu nimisessä luokassa. Valikossa hyödynnetään Button ja Menuitem luokkia, joiden avulla menu nappuloiden ja tekstien luominen ja sijoittelu helpottuu huomattavasti.

Kun menuvalikosta valitaan jokin taso pelattavaksi, ohjelma käynnistää Scene luokan, joka käsittelee kaikki pelin tapahtumat. Scene kutsuu MapLoader luokkaa, joka lataa tekstitiedostosta tason ja luo uuden QGraphicsScenen.

Scene käyttää Derbiili luokkaa pelaajan toimintojen suorittamiseen. Scene ja Derbiili luokassa hyödynnetään myös Signals luokkaa, jossa on määritelty uusia qt -signaaleja. Scenessä tärkein metodi on timerEvent, joka päivittää sceneä ja hoitaa suurimman osan scenessä tapahtuvista asoista.

Animaatiot käsitellään Animation -luokassa. Sen tärkeimpiä ominaisuuksia on erillinen QTimer, sekä load_animations ja next_frame metodit. Kun Animation -luokka on määritelty, voidaan sen metodien avulla vaihtaa animaatioita tai pysäyttää animaatio kokonaan.

Kaikki pelin törmäysfysiikat lasketaan Physics luokassa (sekä pelaajan, että vihollisen). Jokaisella liikkuvalla asialla on oma physics olio, joka on määritelty sen koon ja painon mukaan halutulla tavalla. Olennaisimmat metodit ovat gravity, joka laskee pudotusnopeutta, sekä check_collisions_x ja check_collisions_y, jotka tutkivat törmäyksiä eri suunnissa.

Pelin kaikki QGraphicsItemit on luokiteltu eri paketteihin. Olennaisimmat ovat Creatures, Blocks ja Items paketit. Jokaisessa paketissa on määritelty kaikille siihen kuuluville olioille jokin yläluokka, jonka kaikki muut paketissa olevat oliot perivät. Tämän ansiosta esimerkiksi uusien palikoiden lisääminen on äärimmäisen helppoa, sekä ominaisuuksien lisääminen ja muokkaaminen on onnistuvat silti vaivatta ylikirjoittamalla joitakin yläluokan funktioita.

Mikäli palikalla ei ole mitään erityistä toiminnallisuutta, voidaan luoda vaan Block -luokan palikka, jolle annetaan syötteenä kuvatiedoston sijainti.

Lisäksi on hyödynnetty CONSTANTS moduulia, johon on kirjoitettu erilaisia vakioita. Tämän ansioista, jos haluaa muuttaa esim. painovoimaa, pelaajan nopeutta tai muita ominaisuuksia, niitä tarvitsee muokata vain tässä moduulissa.

Kaikki pelin käyttämät kuvat on lajiteltu Textures -paketin sisään.

6. **Algoritmit**

Ohjelma laskee fysiikoita niin, että pelaajalla on sekä pysty, että vaakasuorassa, jokin nopeus ja ohjelmaa tutkii törmäyksiä eteenpäin nopeuden verran pikseleitä. Ohjelma tutkii kahta pistettä, pysty- ja vaakasuoraan, riippuen pelaajan liikkumissuunnasta. Se käyttää Qt:n ItemAt metodia, jonka avulla laskee, onko pelaaja törmäämässä johonkin palikkaan. Mikäli ItemAt palauttaa jonkin palikan tai esteen, laskee ohjelma pelaajan ja esineen välisen etäisyyden (pelaajan koord. – esineen koord. – pelaajan koko) ja palauttaa sen. Tämän jälkeen pelaaja liikkuu vain etäisyyden verran eikä pääse menemään palikoiden sisälle. Jokaisella pelin esineellä, vihollisella tai palikalla on self.collision attribuutti, jota ohjelma myös tutkii tässä, mikäli se on False, ohjelma antaa pelaajan mennä kyseisen esineen sisälle.

Vaihtoehtona olisi ollut käyttää Qt:n omaa törmäys mekanismiä, joka tutkii törmäyksiä eri QGraphicsItemien välillä. Tämän avulla kuitenkin pelaaja päätyy helposti palikoiden sisälle. Tämä oli mielestäni erittäin alkeellinen ja vaikeasti hyödynnettävissä oleva toteutus.

Pelissä on myös tiedoston lukija, joka lataa pelattavat tasot tekstitiedostoista. Se lukee erikarttoihin liittyvät taustakuvan tiedot, jotka on kirjoitettu #title otsikon alle. Siinä eri tiedot on eritelty ”=” merkillä. #map otsikon alta löytyy itse taso, joka on iso matriisi, jonka läpi ohjelma iteroi ja lisää eri kirjaimilla merkityt palikat sceneen. Ohjelma myös laskee tason koon automaattisesti, kasvattamalla x ja y muuttujia iteroidessa.

Kaikki pelin animaatiot on toteutettu Animation luokan avulla. Jokaiselle animoitavalla QGraphicsItemille määritellään oma Animation olio. Animation luokalle annetaan kansion sijainti, jonka jälkeen se lataa kansion sisällä olevat kuvat Pixmapeiksi. Kuvat täytyy olla järjestetty tietyllä tavalla erilaisten kansioiden sisään, jotta ohjelma lataa ne oikein. Se on kuitenkin tehty todella helpoksi. Sen lisäksi Animation luokalle annetaan parametrina eri framejen välille haluttava viive.

Pelin vihollisilla on metodit, jotka laskevat missä suunnassa pelaaja on suhteessa johonkin viholliseen, eli siis onko pelaaja sen yläpuolella, vieressä tai alapuolella. Sen lisäksi peli laskee pelaajan etäisyyttä vihollisiin. Etäisyys on laskettu pythagoran lauseella ja suunnat tutkimalla pelaajan ja vihollisen koordinaatteja.

7. **Tietorakenteet**

Karttojen tallentamiseen on käytetty tekstitiedostoa, jossa taso on määritelty palikoittain. Tiedostossa on iso pätkä, jossa on erilaisia kirjaimia eroteltuna kaksoispisteessä. Kirjaimet tarkoittavat eri palikoita. Formaatista löytyy tarkemmat ohjeet toisessa tekstitiedostossa.

8. **Tiedostot**

Pelissä on käytetty tekstitiedostoja varastoimaan erilaiset tasot, sekä .png tiedostoja eri grafiikoiden ja animaatioiden talletukseen.

9. **Testaus**

Peliä on testattu pelaamalla, tutkimalla komentorivin antamia virhe ilmoituksia, sekä erilaisilla print -käskyillä. Yksikkötestausta en ole tehnyt ollenkaan, koska se oli erittäin työlästä ja vaikeaa tehdä testiskenaarioita, jossa oikeasti saataisin isompi hyöty verrattuna pelin suorittamiseen ja kokeilemiseen.

10. **Ohjelman tunnetut puutteet ja viat**

    Viat:

    Jos pelaaja osuu tietyllä tavalla piikkipalikan nurkkaan, pelaaja ei aina kuole, vaikka selvästi näkee, että pelaaja osuu ja törmää piikkeihin. Vika johtuu törmäystunnistuksen ja vihollisen vaikutuksen järjestyksestä, pelaaja kerkeää hipaista ja törmätä piikin nurkkaan ja liikkua siinä pois ennen kuin peli kerkeää tunnistaa pelaajan törmänneen piikkiin.

    Peli kaatuu välillä, jos pelaaja on käynyt pelissä ja palaa valikkoon ja yrittää sen jälkeen valita uusia näppäin asetuksia. Tämä ei kuitenkaan tapahdu aina, joka on tehnyt vianmäärityksestä hankalaa.

    Viimeisen tason kruunun animaatio ei ole synkronoitu pelaajan pään liikkeen mukaan. Lisäsin kruunun hyvin loppu vaiheessa ja en kerennyt ratkaisemaan ongelmaa. Ajatuksia oli kuitenkin luoda jonkinlainen synkronointi systeemi Animation -luokkaan, jolloin molemmat animaatiot käyttäisivät samaa ajastinta. Tässä voisi hyödyntää signaaleja, sekä connect metodia.

    Puutteet:

    Animation luokan toteutus olisi voinut olla vielä hieman parempi. Olisi ollut järkevämpää tehdä yhden animaation kaikki kuvat samaan .png tiedostoon. Opin tekemään tämän vasta ihan loppu vaiheessa ja ei ollut aikaa enää ohjelmoida luokkaa uudelleen.

    Pause -luokan olisi voinut myös tehdä QTimerin avulla, joka olisi ollut hieman järkevämpi toteutus. Tämänkin opin liian loppu vaiheessa ja en enää kerinnyt tehdä luokkaa uudelleen.

    Olisin halunnut lisätä Movement -luokan, joka käsittelisi kaikki erilaiset liikkeen, jolloin uusien vihollisten liikkeiden ohjelmointi olisi paljon helpompaa ja koodi olisi muutenkin selkeämpi. Tällainen luokka olisi kannattanut tehdä jo alussa. En halunnut enää tässä vaiheessa lähteä rikkomaan koodia näin perusteellisella muutoksella.

11. **3 parasta ja 3 heikointa kohtaa**

    Hyvät:

    Itse luettelisin pelin parhaiksi puoliksi grafiikat, fysiikat sekä laajennettavuus.

    Olen piirtänyt pelin kaikki grafiikat itse, sekä luonut animaatiot. Olen todella tyytyväinen pelin ulkonäköön.

    Pelin fysiikat ovat mielestäni erinomaiset. Mikäli projektia haluaisi jatkaa, olisi niissä silti vielä hiottavaa. Kuitenkin, pelaajan liike on kiihtyvää joka suunnassa ja maalla ollessa pelaajalla on tietty kitka, jolla sen liike hidastuu. Tokassa tasossa tämä havainnollistuu jääpalikoiden avulla hyvin. Törmäykset eri palikoihin jne. toimii lähes täydellisesti.

    Peli on erittäin varma toiminen. Tämän hetkisessä tilassa pelin laajuuden huomioiden pelissä on suhteellisen vähän bugeja ja bugit ovat aika mitättömiä.

    Huonot:

    Heikkoja kohtia on mielestäni pelissä menu valikko, joka on aika suppea. Se voisi olla paljon graafisesti miellyttävämpi ja paremmin toteutettu.

    Projekti voi vaikuttaa hieman keskeneräiseltä ja kuten TODO-listasta näkee, olisi projektiin vielä paljon lisättävää, jotta sitä voisi täysin valmiiksi sanoa. Siitä huolimatta projekti on erittäin toimiva kokonaisuus jo nyt.

12. **Poikkeamat suunnitelmasta**

    Pelin toteutus poikkeaa aika paljon suunnitelmasta, koska hylkäsin kokonaan vuorikiipeily idean, koska se olikin mielestäni tylsä ja halusin tehdä erilaisen pelin. Muuten kuitenkin erilaisten algoritmien ja tiedostorakenteiden kannalta pelin toteutus on mennyt hyvin suunnitelman mukaan. Tietenkin pieniä muutoksia lukuun ottamatta.

13. **Toteutunut työjärjestys ja aikataulu**

    Aloitin pelin ohjelmoinnin hyvissä ajoin, jo ennen projektin aloitus tapaamista. Siinä vaiheessa olin saanut peliin jo luotua jokseenkin toimivat fysiikat ja muutaman palikan. Pelin tekeminen on edennyt aikataulun mukaan, mutta joitakin ominaisuuksia olen joutunut karsimaan ajan puutteen vuoksi. Peliin on kuitenkin käytetty tasaisesti joka viikko huomattava määrä aikaa ja se näkyy pelin laajuudessa.

14. **Arvio lopputuloksesta**

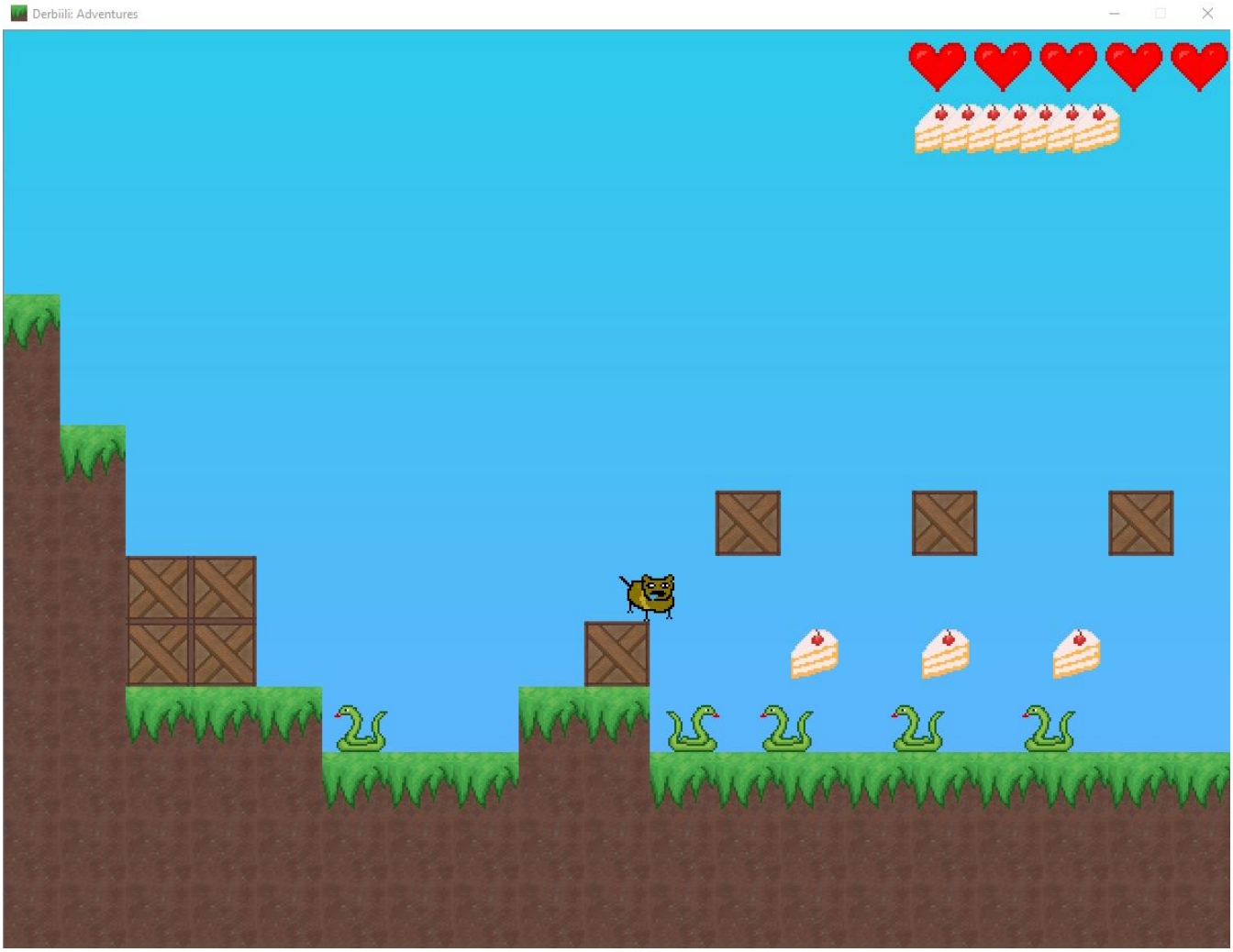
    Kokonaisuudessaan olen tyytyväinen peliin. Se on mielestä erittäin toimiva ja suhteellisen laaja kokonaisuus. Olen pyrkinyt tekemään ratkaisuja, joiden avulla pelin laajennettavuus olisi mahdollisimman helppoa. Siinä olen onnistunut erittäin hyvin vihollisten, animaatioiden, palikoiden ja tavaroiden kanssa. Fysiikoiden ja tarkemmin liikkumisen suhteen niistä voisi tehdä helpompi käyttöiset ja laajennettavat, sekä toimivuus eri kokoisille vihollisille voisi olla parempi. Jos jatkaisin projektia, tekisin fysiikoista helpompi käyttöiset ja peliin voisi lisätä paljon erilaisia tasoja, luovuus vain rajana.

15. **Viitteet**

    Olen käyttänyt apuna pääasiassa Qt:n omaa dokumentaatiota. Sen lisäksi olen hakenut eri nettisivuilta pyqt kohtaisia ohjeita tarkempiin ongelmiin. Aluksi myös katsoin hieman apua kurssimateriaaleista, mutta ne olivat varsin suppeat projektin tekemiseen.

16. **Liitteet**

    Seuraavalla sivulla on muutama kuva pelistä.


Derbiili: Adventures