Tekninen suunnitelma (Father Long Legs)

Jonathan Leinola, 477329

1. **Ohjelman rakennesuunnitelma**

   Ohjelman rakennesuunnitelma jakautuu seuraaviin luokkiin:

   - Player
   - Platform
   - Level
   - Spritesheet

   Player luokassa määritellään esimerkiksi mitä kohtaa ruutua pelaaja kontrolloi. Tässä tapauksessa pelaaja on ruudun alareunassa, ja kun liikkuminen tapahtuu, päivitetään maailmaa sen mukaan. Player luokassa keskeiset metodit ovat pelaajan liikkuminen (update), pelaajan hyppääminen (jump) ja gravitaation laskeminen pelaajalle (grav). Platform-luokassa määritetään taso, jolle pelaaja pystyy hyppäämään.

   Level-luokka tarvitaan eri kenttien määrittelyyn. Sen keskeisimpiä metodeja on päivittää kenttä (update), piirtää kaikki tarvittavat asiat (draw) ja ”maailman siirtäminen” pelaajaluokan mukaisesti. Update-metodia tarvitaan siihen, että tarvittavat tiedot tulevat päivitetyksi ruudulle. Kun luodaan useita kenttiä, nämä kentät perivät Level-luokan tähän tapaan:

   ```
   Class Level1(level):
   ```

   Spritesheet-luokan tarkoituksena on lukea kuvatiedostoja ja ottaa sieltä tarvittavat kuvat päivittääkseen hahmoa, tai kenttää. Spritesheet-luokan tärkein metodi on hakea kuva (getimage), jossa kuva haetaan pelin kuvakansiosta ja iso kuva paloitellaan oikean kokoisiksi. Nämä kuvat asetetaan sitten esimerkiksi Player-luokassa hahmon kuvaksi.

   Perinnästä vielä, koska ohjelmassa käytetään pygame kirjastoa, perivät luokat player ja platform luokan pygame.sprite, joka on yksinkertainen luokka näkyvien peliobjektien luomiseen.

2. **Käyttötapauskuvaus**

   Ohjelman käynnistämisestä ensimmäisen kentän suorittamiseen:

   Ensimmäisenä käyttäjälle tulee graafisessa käyttöliittymässä vastaan aloitusnäyttö. Käyttäjä voi valita aloittaako hän pelin vai haluaako hän katsoa tilastoja. Tässä tapauksessa käyttäjä valitsee pelin aloituksen nuolinäppäimiä ja Enter-painiketta käyttämällä. Tässä vaiheessa ohjelma on mm. initalisoinut mixerin (ääniä varten), pelin koon ja pelaajan. Aloitusnäyttö on ensimmäinen looppi ennen ohjelman päälooppia, jonne päästään, kun valitaan aloita peli.

   Peli on alkanut ja kaikki tarvittavat tiedot ovat initalisoitu tässä vaiheessa. Esimerkiksi hahmon liikkumiseen tarvittavat kuvat/spritet. Pelaajan on tarkoitus edetä kentässä väistellen objekteja ja suoriutua tehtävästä mahdollisimman nopeasti. Pääloopissa tapahtuu kaikki oleellinen. Jos pelaaja liikkuu niin päälooppi rekisteröi sen ja liikuttaa sen mukaan kentän taustaa. Ja jos pelaaja pääsee ensimmäisen tason läpi, niin siirrytään seuraavaan tasoon. Jos tasoja ei ole suoritettu tarpeeksi nopeasti tai suoritetaan tasot asetetun ajan mukaisesti, siirrytään takaisin aloitusnäyttöön.

3. Algoritmit

Ohjelmassa tarvitaan painovoima, jotta hahmo pystyy hyppimään ja liikkumaan. Todellisuudessa painovoima on kiihtyvää liikettä, mutta asioita yksinkertaistaakseen tässä tapauksessa painovoima toteutetaan nopeuden lisäyksellä ja vähennyksellä. Tällä tavalla vähennetään prosessin laskentatehoa.

4. Tietorakenteet

Ohjelmassa tarvitaan useita listoja ja yksi esimerkki niistä on matriisien esittämistä varten. Matriiseissa voi olla esimerkiksi kenttien luomiseen tarvittavat esteet, jotka luodaan seuraavanlaisesti:

level = [[50, 70, 500, 500], [x, x, x, x],jne.]

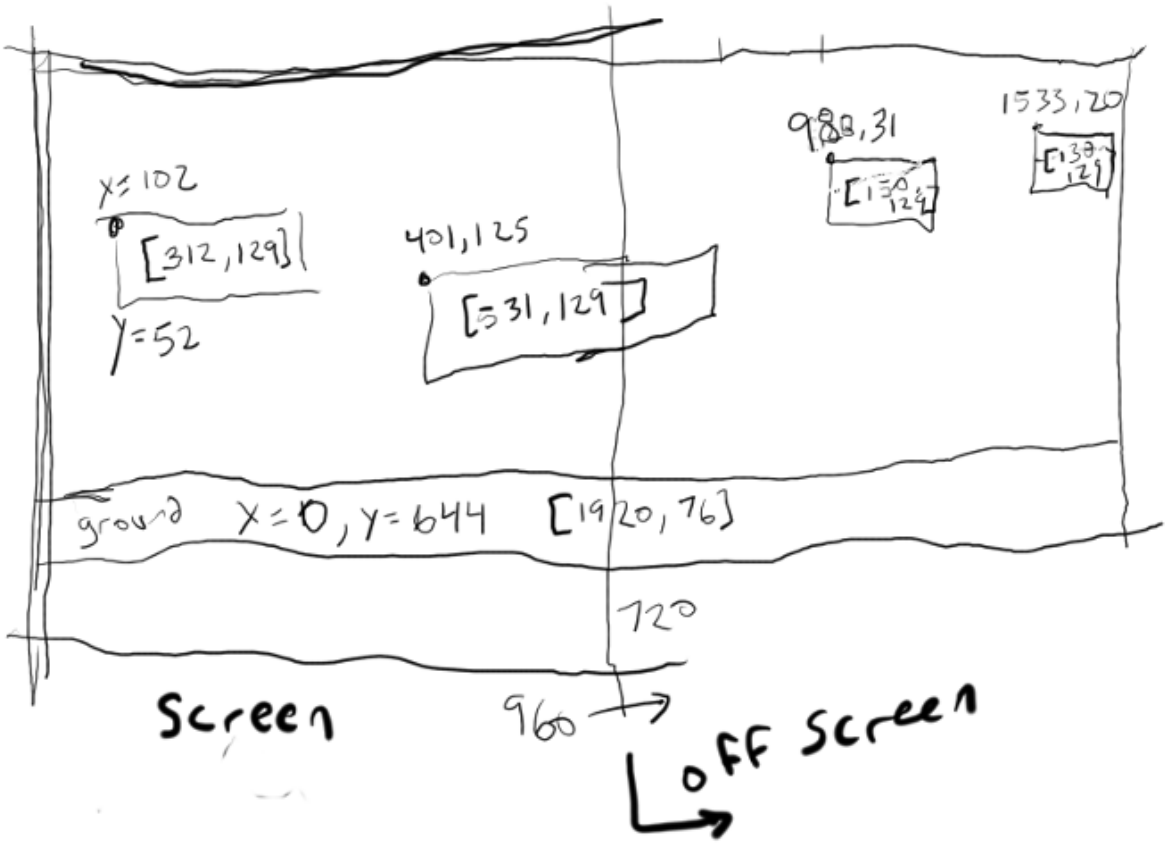

Figure 1 esimerkki kentästä ja sen objekteista

Listan ensimmäinen alkio kertoo esteen leveyden, korkeuden,sekä x- ja y-koordinaatit. Tasojen suunnittelu on tietysti tällä tavalla hieman raskauttavaa, siksi kenttäeditori olisi jatkossa melko kätevä. Sen lisäksi, esimerkiksi pygamessa monikulmioiden pygame.draw.polygon tai suorakulmioiden pygame.draw.rect piirtämisen annetaan parametrina lista pisteitä, jotka ilmaisevat x- ja y-koordinaatteina mihin kohtaan näyttöä monikulmio piirretään.

5. Aikataulu

Työ jakautuu eri osiin, jotka etenevät seuraavien vaiheiden mukaan:

1. Vaihe 2-4h:
   Aloitusnäytön ohjelmointi ja implementointi.
2. Vaihe 6-8h:
   Player-luokan, platform-luokan ja level-luokan ohjelmointi:
3. Vaihe 8h:
   Kenttien luominen (max 2 kenttää)
4. Vaihe 4-6 h:
   Spritesheet luokan ohjelmointi
5. Vaihe 4-8h:
   Spritejen luominen
6. Viimeistelyä 8-12h
   Peliäänien lisäys
   Yksikkötestausta
   Yleisilmeen parantelua

Eri vaiheiden mukaan työtunteja on yhteensä noin 32-46h, joka tekee keskimääräinen n. 10h työtunteja per viikko seuraavan 4 viikon ajan. Viikko suunnitelma:

1. Viikko:
   Vaihe 1 ja 2
2. Viikko
   Vaihe 3 ja 4
3. Viikko
   Vaihe 4 ja 5
4. Viikko
   Vaihe 6

Checkpoint on välillä 18.3-31.3, joten siihen mennessä tämän suunnitelman mukaisesti ollaan vaiheessa 4-6. Lopullinen projektidemo on viimeistään 26.4, joten viimeistelyvaiheeseen pitäisi jäädä hyvin aikaa.

6. **Yksikkötestaussuunnitelma**

   Ohjelmassa keskeisimpiä metodeja ovat gravitaation laskeminen, hahmon hyppääminen ja liikkuminen ja tietysti kentän piirtäminen. Näiden lisäksi hahmon liikkumisen rajaaminen ja törmäyksentarkistus on syytä tarkistaa toimivaksi.

   Gravitaatiota laskiessa on syytä tarkastaa, jos hahmo on maassa niin sen pitäisi palauttaa arvon 0. Muussa tapauksessa sen pitäisi palauttaa arvon, joka on erisuuri kuin 0.

   Hahmon liikkumisen rajauksen testaaminen testataan siten, että verrataan, onko hahmon neliö päivitetyn tason ulkopuolella. Mikäli hahmo on tason ulkopuolella, palautetaan error exception.

   Törmäyksentarkistus tapahtuu siten, että verrataan pelaajan neliön nurkkaa kentän tasojen nurkkien pisteisiin. Jos pisteet ovat päällekkäin, silloin törmäyksentarkistus palauttaa error exceptionin.

7. **Kirjallisuusviitteet ja linkit**

   https://fi.wikipedia.org/wiki/Sprite-grafiikka

   https://en.wikipedia.org/wiki/Game_testing

   https://www.theseus.fi/bitstream/handle/10024/21239/kuisma_joentakanen.pdf?sequence=1

   https://linkki.github.io/python2017/pygame.html

   https://www.cs.helsinki.fi/group/linkki/materiaali/peliohjelmointi/lista.html

   https://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Sprite

   https://opensource.com/article/18/7/put-platforms-python-game