Yleissuunnitelma, tasohyppely-peli

Lauri Westerholm, 530868

Automaatio 2. vuosikurssi, 16.2.2018

1. **Yleiskuvaus**

Peli: Tasohyppely

Toteutetaan tasohyppely peli, jossa pelaaja ohjaa hahmoa kentän läpi hyppimällä erilaisten tasojen päälle.

- Graafinen käyttöliittymä (pelaaja ohjaa pelihahmoa mm. nuolinäppäimillä).
    - Aloitusnäyttö, josta pääsee varsinaiseen peliin painamalla Start Game -nappulaa
    - Lopetusnäyttö, joka kertoo pelin tuloksen (voitto vai häviö). Siirtyminen alkuun painamalla Continue-nappia
- Toimiva törmäyksen tunnistus. Ohjelman täytyy tunnistaa, kun kaksi tai useampi objekti törmäävät. Esimerkiksi, jos pelaajan hahmo juoksee estettä päin, niin ohjelman tulisi huomata tämä ja estää pelihahmon eteneminen esteen läpi.
- Hyppymekaniikka, objektien päälle pystyy hyppäämään
- Vihollisia (ei varsinaista tekoälyä, liikkuvat edes takaisin vasemmalta oikeallle)
- Pelaaja voittaa, kun läpäisee kentän (hyppää/osuu maaliobjektiin). Häviäminen: pelaaja osuu objektiin, joka ns. tiputtaa pelihahmon pelikentältä tai vihollinen osuu pelaajaan
- Valmis kenttä(t). Kentät etenevät lähtökohtaisesti vasemmalta oikealle.
- Pelihahmon ja vihollisten grafiikkana yksinkertaisia kuvia (esim. jpeg muodossa)
- Kentät tiedostomuodossa (tekstitiedosto), josta sitten luetaan pelikentäksi

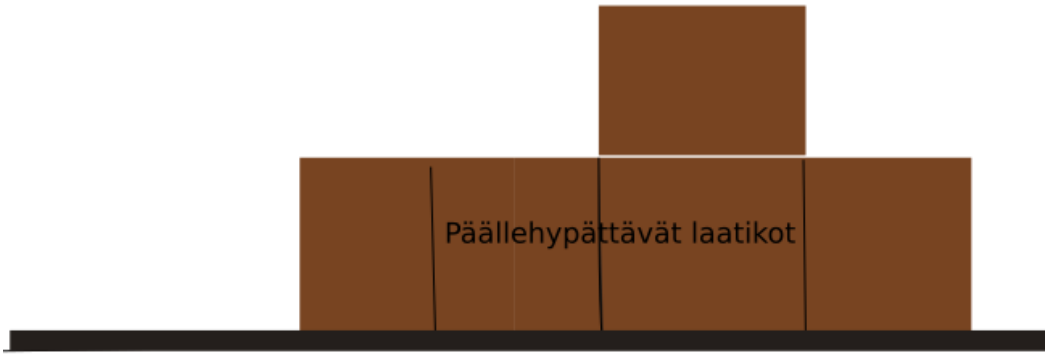
Vaikeustaso: vaikea (keskivaikea)

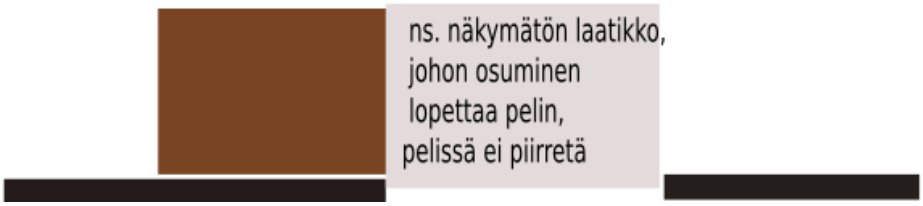
2. K**äyttöliittymän luonnos**

Ohjelmassa on ensinäkin aloitusnäyttö, josta käynnistetään Start Game -napilla varsinainen peli.

Ohjelma saa käyttäjältä syötteensä näppäimistön left arrow (vaihtoehtoisesti A), right arrow (vaihtoehtoisesti D) ja space painalluksina. (Aloitusnäytössä peli käynnistetään hiiren painalluksella.) Pelin saa suljettua Esc -painikkeella.

Käyttäjän näppäinpainallusten perusteella pelihahmoa (graafista objektia) liikutetaan pelilaudalla. Nuolilla sivulle ja spacella hypätään.





Yllä olevassa kuvassa hahmotelma siitä, miltä tasohyppelyn kenttä laatikko-objektien osalta näyttää pelaajalle.

3. **Tiedostot ja tiedostoformaatit**

Pelin kentät säilytetään tiedostossa. Kentän tiedot ovat tekstitiedostossa, jolla on tietty rakenne ja joista olennaiset tiedot pelikentästä luetaan pelikenttäolioon.

```
%PlayField
Width,50

%Player
Location,0;0

%Enemies
10;0;s, 18;3;s, 32;0;f, 37;2;s

%Visible_barrels
3;0, 4;0, 5;0, 6;0, 6;1, 7;0, 7;1, 12;0, 13;0, 13;1, 14;0, 14;1, 14;2, 15;0,
15;1, 15;2, 16;0, 16;1, 16;2, 16;3, 17;0, 17;1, 17;2, 18;0, 18;1, 18;2, 19;0,
19;1, 19;2, 20;0, 20;1, 20;2, 20;3, 21;0, 21;1, 21;2, 22;0, 22;1, 23;0, 27;0,
28;0, 30;0, 33;0, 34;0, 34;1, 35;0, 35;1, 35;2, 36;0, 36;1, 36;2, 37;0, 37;1,
38;0, 38;1, 39;0, 39;1, 39;2, 40;0, 40;1, 40;2, 40;3, 41;0, 41;1, 41;2, 41;3,
42;0, 42;1, 43;0, 46;0, 47;0, 48;0, 48;1, 49;0

%Invisible_barrels
25, 44

%Finish
49;1
```

Yllä olevassa kuvassa tekstitiedoston rakenne, joka sisältää pelinkentän. (Huomiota: näkyvien laatikoiden ja vihollisten koordinaatit esitetty muodossa x; y. Vihollisilla vielä nopeusmäärite (s = slow, f = fast). Elementit pitää olla tiedostossa tässä järjestyksessä, muuten ohjelma ei voi muodostaa pelikenttää). Tiettyjen elementtien puuttuminen (pois lukien pelaaja ja maali sekä leveys) ei haittaa.

Pelaajahahmo sekä viholliset luetaan suorakaiteen muotoisiksi piirrettäviksi objekteiksi kuvamuodosta (png).

 vihollinen

 pelihahmo

4. **Järjestelmätestaussuunnitelma**

Testaus tulee tässä tapauksessa jakaa osiin: yksikkötestata erikseen graafisen käyttöliittymän toimintaa, pelilaudan muodostamista ja objektien pelikentälle asettamiseen sekä liikuttamiseen käytettäviä metodeja. Sen jälkeen, kun koodi on kokonaisuudessaan valmis, tulee testata varsinaisen pelin toimintaa.

Ohjelma lähtee liikkeelle pelikentän lukemisesta tiedostosta ja on kriittistä, että tiedoston lukeminen onnistuu. Tässä tapauksessa käyttäjä ei kuitenkaan syötä tiedostoa, vaan valmiiseen peliin kuuluu osana kentät sisältävät tiedostot. Laajennettavuutta varten tulee kuitenkin lisätä tiettyjä varomekanismeja tiedoston lukemiseen, ja näiden varomekanismien toimintaa (pelikentäksi yritetään lukea virheellisen muotoinen tiedosto, sisältää esimerkiksi objektien paikkoja pelialueen ulkopuolelta) tulee jossain määrin yksikkötestata. Tässä tapauksessa yksittäinen testi pitkälti riittää, koska tällaista tilannetta ei pitäisi normaalikäytössä tulla ikinä vastaan, koska pelaaja ei syötä tiedostoja.

Sen sijaan pelihahmon liikutteluun liittyvät toiminnot vaativat huomattavasti yksikkötestausta. Ensinäkin pitää varmistua, että hahmon ohjaamiseen käytettävien näppäinten painallukset rekisteröityvät ja saadaan välitettyä eteenpäin. Toisaalta taas muiden näppäinten (pois lukien esc, joka sulkee pelin) painaminen ei saa vaikuttaa peliin.

Toisekseen pitää testata, että pelihahmo ja viholliset ovat pelikentällä oikeassa paikassa sen jälkeen, kun niitä on liikutettu. Tämä on välttämätöntä, jotta grafiikka saadaan piirrettyä oikein. Pelihahmoon liittyvien hyppy/pudotus metodien oikea toiminta on myös välttämätöntä (pitää testata, että pelihahmo todellakin pystyy hyppäämään objektien päälle). Hahmo ei myöskään saa pudota objektien läpi tai pystyä kulkemaan läpi objekteista. Collosion-detectionia pitää siis yksikkötestata laajasti.

Piirtämisen osalta tulee testata, että objektit todella piirtyvät oikeisiin paikkoihin ja että hypyn/pudotuksen piirtäminen toimii järkevästi eli on sulava. Tässä keskeistä tulevat olemaan objektien koot pikseleinä suhteessa toisiinsa ja piirtäminen vaiheittain.

Tämän lisäksi tulee testata, että siirtymät aloitusruudun, varsinaisen pelin ja lopetusruudun välillä toimivat. Kokonaan valmis peli on sellainen, että käyttäjä ei saa pystyä rikkomaan sitä millään näppäimistön inputilla. Peli ei tietenkään saa itsessään kaatua, päätyi tilanne pelilaudalla mihin tahansa mahdolliseen tilanteeseen. Pelihahmon ohjauksen pitää siis esimerkiksi sisältää esto hahmon ohjaamiseen pelikentän ulkopuolelle: Pelihahmo on jo valmiiksi kentän vasemmassa reunassa. Käyttäjä ei saa onnistua ohjaamaan hahmoa enää lisää vasemmalle.

Lopullisen ohjelman ehyettä on helpointa testata pelaamalla peliä ja kokeilemalla, että se toimii kaikissa erikoisemmissakin tilanteissa oikein. Tässä tapauksessa on melko mahdotonta tehdä valmista testiohjelmaa, joka kattaisi kaikkia mahdollisia pelaajan liikkeitä. Onkin tärkeää, että katetaan kaikki mahdolliset pelin lopputulokset ja sijainnit pelikentällä - mikään niistä ei saisi aiheuttaa pelin kaatumista.

Lisäksi kokonaisen pelin responsiivisuutta pitää testata. Pelin grafiikan piirtäminen (ja graafisen ikkunan ylläpito sisältäen näppäin inputin taltioinnin) on erillään objektien piirtoa varten tapahtuvasta laskennasta. Näin olen laskennan/piirtämisen pitää tapahtua sopivassa suhteessa, muuten peli jähmettyy.