Turku Horror

Harjoitustyö 3 + 4

Kesäkuu 2008

Pekka Salmela

64749

pisalm@utu.fi

Sisällysluettelo

1. Tehtävän kuvaus ja analysointi

Arkham Horror on alunperin Chaosiumin vuonna 1987 ja Fantasy Flight Gamesin uudistettuna versiona vuonna 2005 julkaisema lautapeli, joka perustuu temaattisesti kauhukirjailija Howard Phillips Lovecraftin teoksiin. Pelissä 1-8 pelaaja ottavat kukin hahmokseen tietyillä kyvyillä ja esineillä varustetun tutkijan, ja yhdessä he pyrkivät estämään ikivanhaa pahuutta edustavan Suuren Muinaisen olennon tunkeutumisen maailmaan. Pelin tapahtumapaikkana toimii fiktionaalinen Arkhamin kaupunki Massachusettsissa, USA:ssa.

Tämän harjoitustyön tarkoituksena on mallintaa Arkahm Horrorin pelimekaniikan keskeisimpiä osia käyttäen Python-ohjelmointikieltä. Pyrkimyksenä ei ole suoraan siirtää Arkham Horror -peliä digitaaliseen muotoon, vaan luoda ohjelmallinen perusta pienimuotoisemmalle, *Turku Horror* -nimiselle PC-pelille, jonka sääntömekaniikka mukailee Arkham Horrorin sääntöjä, mutta joka sijoittuu Turun kaupunkiin nykyaikaisessa Suomessa.

Koska Arkham Horrorin säännöt ovat laajuudeltaan varsin mittavat, ja niihin sisältyy hyvin paljon erilaisia yksityiskohtia, on tässä työssä keskitytty pelin tärkeimpien elementtien ja keskeisimpien sääntöjen mallintamiseen. Tällaisia ovat pelihahmon ja pelilaudan mallintaminen, liikkuminen pelilaudalla, hahmon kykyarvojen säätäminen ja erilaisten kykytarkistusten tekeminen.

Tehtävän ohjelmoinnissa on noudatettu testivetoista ohjelmointia. Tämä tarkoittaa sitä, että.ennen minkään pelin elementtiä kuvaavan moduulin tai metodin kirjoittamista sitä varten on kirjoitettu testi, joka testaa kyseisen moduulin tai metodin toiminnallisuutta. Harjoitustyön tarkoituksena onkin ollut harjoitella sekä Python-ohjelmointikielen käyttöä että testivetoisen ohjelmoinnin hyödyntämistä.

2. Ratkaisuperiaate

Tehtävän ratkaisu perustuu kahteen ohjelmapakettiin ja yhteen paketeista erilliseen moduuliin.

- Paketti `src` sisältää pelin eri elementtejä mallintavia moduuleita. Tällaisia elementtejä ovat pelihahmo (tutkija), pelilauta, paikat pelilaudalla sekä hahmojen suorittamat kykytarkistukset.
- Paketti `data` sisältää moduuleja, jotka sisältävät kovakoodattua tietoa Turku Horror -pelin pelilaudasta, hahmoista yms.

- Moduuli `game` käynnistää pääohjelman, joka käyttää yllä kuvattuja paketteja mahdollistaa pelien elementtien ja sääntöjen toiminnan kokeilemisen käytännössä.

3. Ohjelman ja sen osien kuvaus

3.1 Paketti src

3.1.1 area.py

Moduuli, joka mallintaa katualuetta, joka voi olla yhteydessä yhteen tai useampaan kohteeseen (ks. moduuli `location`) ja yhteen tai useampaan muuhun katualueeseen. Moduuli sisältää metodit kyseiseen alueeseen yhteydessä olevien kohteiden ja alueiden lisäämiseen. Lisäksi moduuli sisältää metodit, joilla voidaan hakea tietyn askelmäärän päässä kyseisestä alueesta olevat kohteet ja alueet sekä laskea tarvittavien askelten määrä kuljettaessa kyseiseltä alueelta annettuun kohteeseen tai alueelle.

3.1.2 board.py

Moduuli, joka mallintaa pelilautaa. Pelilauta sisältää toisiinsa yhteydessä olevia kohteita (ks. moduuli `location`) ja alueita (ks. moduuli `area`). Moduuli sisältää metodit kohteiden ja alueiden lisäämiseen laudalle.

3.1.3 investigator.py

Moduuli, joka mallintaa tutkijaa (eli pelihahmoa), jolla on lukuisia attribuutteja. Tutkijan tietoihin kuuluvat nimi, ammatti, maksimaallinen ja nykyinen fyysinen terveys (stamina), maksimaallinen ja nykyinen mielenterveys (sanity), keskittyminen (focus), nykyinen sijainti ja rahavarojen määrä. Lisäksi tutkijalla on kuusi kykyä (speed, sneak, fight, will, lore, luck), joita kaikkia vastaa kyvyn nykyistä tilaa kuvaava lukuarvo, sekä lista, joka sisältää neljä lukuarvoa, jotka kyseinen kyky voi saada. Moduuli sisältää myös metodit attribuuttien muokkaamista varten.

3.1.4 location.py

Moduuli, joka mallintaa maantieteellistä kohdetta (kuten kirkko, hotelli jne) pelilaudalla. Kohde on yhteydessä ainoastaan yhteen alueeseen (ks. moduuli `area`).

3.1.5 skillcheck.py

Moduuli, joka sisältää metodeja kykytarkistuksen tekemiseen. Kykytarkistukseen vaikuttaa tutkijan

taito sekä tarkistukseen liittyvä tehtävämuuttuja (modifier) ja vaikeustaso (difficulty). Kykytarkistuksessa kyseisen taidon lukuarvosta vähennetään tehtävämuuttuja ja saatu luku kertoo kuinka montaa kuusitahoista noppaa tutkija saa heittää. Tarkoituksena on saada noppatulos viisi tai kuusi vähintään niin monta kertaa kuin vaikeustasoa kuvaava luku osoittaa. Mikäli tämä onnistuu, tarkistus on läpäisty, muuten kyseessä on epäonnistuminen. `skillcheck`-moduuli sisältää metodit kykytarkistuksen suorittamiseen ja sekä heittotuloksen että onnistumisen/epäonnistumisen määrittelyyn.

3.2 Paketti data

3.2.1 characters.py

Moduuli, jonka on tarkoitus sisältää pelissä käytössä olevien tutkijoiden eli hahmojen tiedot. Tässä vaiheessa moduuli sisältää vain yhden metodin, joka palauttaa pääohjelmassa käytettävän esimerkkihahmon, professori Veijo Hietalan tiedot.

3.2.2 logo.py

Moduuli, joka sisältää vain yhden metodin, joka puolestaan palauttaa listan, jonka alkioiden tulostaminen allekkaisille riveille muodostaa pelin logon, joka esitetään peliä käynnistettäessä.

3.2.3 map.py

Moduuli, joka sisältää vain yhden metodin, joka puolestaan palauttaa listamuotoisena tekstimuotoisen esityksen kartasta, joka esittää Turkua.

3.2.4 turkuboard.py

Moduuli, joka sisältää vain yhden metodin, joka puolestaan rakentaa neljästä alueesta ja 12 kohteesta koostuvan, `board`-moduulin mukaisen ja Turkua kuvaavan pelilaudan ja palauttaa sen.

3.3 Pääohjelma

3.3.1 game.py

Moduuli, joka mahdollistaa Turku Horror -pelin pelaamisen niiltä osin, kuin se toteutettujen toiminnallisuuksien osalta on mahdollista. Peli kuljettaa pelaajaa kolmen eri vaiheen läpi, ja antaa kussakin vaiheessa mahdollisuuden tehdä kyseiseen vaiheeseen liittyviä valintoja. Koska pelin

sisällöstä on toteutettu vain pieni osa, ei pelaaminen ole erityisen mielekästä, mutta se antaa kuitenkin mahdollisuuden testata toteutettuja elementtejä käytännössä.

Ensimmäisessä vaiheessa (upkeep) pelaaja voi säätää kuutta kykyarvoaan haluamaansa suuntaan. Säätämisen määrää hahmon keskittymisarvo, joka kertoo sen, kuinka monta ”pykälää” pelaaja voi yhteensä kykyarvoja säätää. Jokainen kyky on paritettu yhteen jonkun toisen kyvyn kanssa (parit ovat speed ja sneak, fight ja will ja lore ja luck) ja kun toista parista nostaa pykälällä, toinen vastaavasti laskee pykälällä.

Toisessa vaiheessa (move) pelaaja voi liikkua pelilaudalla, jonka kartan hän voi myös halutessaan saada näkyviin. Liikkumisen määrä riippuu hahmon speed-kyvyn arvosta, joka määrää, kuinka monta liikkumispistettä hahmolla on käytettävissään. Jokainen siirtymä kohteen ja alueen tai kahden alueen välillä maksaa yhden liikkumispisteen. Pelaaja voi lopettaa liikkumisen koska vain haluaa.

Kolmannessa vaiheessa tapahtuu jotain vain, jos hahmo on liikkumisen lopuksi jossain kohteessa (ei siis millään alueella). Tällöin hänelle esitetään vapaaehtoinen haaste, jonka vastaanottaessaan hänen tulee tehdä kykytarkistus. Jos hän läpäisee tarkistuksen, hän saa palkinnon. Mikäli hän epäonnistuu, kärsii hän rangaistuksen.

Peli päättyy jos pelaaja valitsee valikosta pelistä poistumisen tai jos pelaajan hahmon fyysinen terveys tai mielenterveys tippuu nollaan.

Pelin käyttöliittymän toteuttamiseen on käytetty Pythonin `curses`-nimistä kirjastoa, joka mahdollistaa erilaisten elementtien sijoittelun ja esittämisen pääte-emulaattori-ikkunassa. Kyseessä on suhteellisen yksinkertainen ja helppo menetelmä luoda yksinkertaisia tekstipohjaisia käyttöliittymiä, mutta esim. virheidenkäsittelyn osalta sen käyttöönotto loi myös ongelmia.

4. Testaus

Koska harjoitustyön yhtenä tarkoituksena oli harjoitella testivetoista ohjelmointia, on jokaista `src`-paketin moduulia varten kirjoitettu testimoduuli, joka testaa kattavasti kyseisen moduulin toiminnallisuutta. Kaikki testit on kirjoitettu ennen kyseisen toiminnallisuuden kirjoittamista ja

pyrkimyksenä on ollut, että mitään `src`-pakettiin kuuluvaa koodia ei ole kirjoitettu ennen kuin sitä testaava testitapaus on olemassa.

Testien tekemiseen on käytetty Pythonin unittest-kirjastoa. Jokaista moduulia varten on oma testimoduulinsa, joka sisältää yhden tai useamman testiluokan, jotka puolestaan sisältävät varsinaiset testimetodit. Jokainen testimetodi on erillinen kaikista muista, ja jokaisen toiminnallisuuden testaamista varten on pyritty kirjoittamaan oma metodinsa. Moduuli `alltests` kokoaa kaikkien testimoduulien sisältämät testit ja ajaa ne läpi kerralla. Ajamalla kaikki testit läpi aina kun testattaviin luokkiin on tehty jotain muutoksia on pystytty havaitsemaan heti, jos jokin muutos on saanut aikaan jonkun moduulin rikkoutumisen.

Paketin data moduuleja varten ei ole olemassa testejä, koska kyseisen paketin moduulit toimivat lähinnä säilytyspaikkana erilaisille elementeille, joita pääohjelma käyttää pelin alustamiseen. Itse pääohjelmaa on ajettu toistuvasti aina uusia ominaisuuksia lisättäessä jolloin erilaiset virheet on havaittu ajoissa. Myöskään pääohjelmaa varten ei ole olemassa erillistä testimoduulia.

5. Käyttöohje

Ohjelma on kirjoitettu ja testattu käyttäen Python-kielen versiota 2.5, Ubuntu Linux -käyttöjärjestelää sekä Gnome Terminal -pääte-emulaattoria. Ohjelman toimintaa ei ole varmistettu muilla käyttöjärjestelmillä tai vanhemmilla Python-kielen versioilla.

Ohjelman toimimiseksi käyttöjärjestelmän PYTHONPATH-polkuun tulee lisätä polku ohjelman pääkansioon, esim. /home/user/horror/. Tämän jälkeen pääohjelma voidaan käynnistää komentoriviltä komennolla

```
python horror/game.py
```

Testiohjelma puolestaan käynnistyy komennolla

```
python test/alltests.py
```

Mikäli ohjelma kaatuu virheilmoitukseen

```
error: addstr() returned ERR
```

voi se johtua siitä, että ikkuna, jossa ohjelmaa ajetaan, on niin pieni, että kaikki käyttöliittymän piirtämät elementit eivät mahdu siihen, mikä puolestaan johtaa `curses`-kirjaston `addstr`-metodin kaatumiseen. Ikkunan suurentaminen voi auttaa asiaa.