1. **Henkilötiedot**

   Otsikko: Pet Wars: Rise of Fluffy
   Tekijän nimi: Rasmus Kull
   Opiskelijanumero: 884433
   Koulutusohjelma: CS-A1121
   Vuosikurssi: 2023
   Päiväys: 12.05.2023

2. **Yleiskuvaus ja vaikeustaso**

   Strategiapeli

   Tein vuoropohjaisen strategiapelin tekoälykkäällä tietokonevastustajalla. Käytin inspiraationa XCOM -pelisarjaa. Käytin alustavasti pohjana harjoituksissa olevaa robottimaailmaa, minkä takia monet käyttämistäni luokista ovat hyvin samankaltaisia kuin robottimaailman luokat. Tekoäly tarkoittaa, että viholliset eivät toimi pelkän satunnaisuuden perusteella, vaan jollain tavalla sääntöjensä perusteella yrittävät toimia tehokkaasti. Tässä tapauksessa tekoäly valitsee toimintansa useamman if-lauseen perusteella. Jos lemmikki "näkee" vihollisen ja jos itsellään on yli puolet elämää jäljellä, sekä vähintään 5 manaa, menee lemmikki kohteensa luokse ja hyökkää. Hyökkäys riippuu manan määrästä ja kohteen elämästä:
   - jos kohteella on yli 40% elämästä jäljellä ja lemmikillä tarpeeksi manaa, tekee tämä "heavy attack":n
   - muussa tapauksessa tavallisen hyökkäyksen
   - muuten jos lemmikillä vähintään 10 manaa ja on haavoittunut, parantaa lemmikki itsensä ja liikkuu pakoon sattumanvaraiseen ruutuun
   - muuten jos lemmikin mana on alle 10, palauttaa lemmikki manansa täyteen ja ei liiku
   - jos mikään muu aiempi ehdoista ei täyty, liikkuu lemmikki sattumanvaraisesti

   Pelin synopsis on seuraavanlainen: ” The Great Pet War: A long time ago, the various pet species lived in peace with each other until a catastrophic event (Humanity’s extinction) caused them to turn on each other. The Great Pet War lasted for years and decimated the pet population. Eventually, the pets formed factions based on their species, and the war ended with the formation of the Pet Council, a governing body that seeks to maintain peace between the factions. For centuries the pet factions lived in peace under the watchful eye of the Pet Council. That is, until today. A scorned and disgraced general of the Alliance of Cats, Fluffy, created a weapon of mass-destruction, capable of mind control. He used it to enact vengeance upon all of pet-kind. You are a pet finding themselves to be the only one immune to the evil power of mind control. It is up to you to face off against the tyrant and save all pet-kind, building a stronger team along the way.”

Pelin aikana pääset pelastamaan lemmikkejä ja lisäämään ne osaksi tiimiäsi. Eri eläimillä on eri heíkkouksia ja vahvuuksia. Lemmikeillä on myös erilaisia hyökkäyksiä. Pelissä on erilaisia tasoja ja ympäristöjä, joissa joudut taistelemaan vihollislemmikkejä vastaan. Näissä ympäristöissä on esteitä, jotka lemmikkien on kierrettävä. Olen tätä varten luonut algoritmin, joka laskee lemmikeille mahdolliset siirrot, ottaen huomioon esteet. Poikkeuksena tästä ovat linnut, sillä ne voivat lentää esteiden ylitse ja voivat halutessaan laskeutua niiden päälle.

Olen luonut kaiken kaikkiaan viisi eläinluokkaa (Cat, Dog, Rodent, Reptile, Bird), joilla on omat kyvykkyytensä. Olen myös luonut grafiikat jokaista eläinluokkaa varten. Lisäksi olen luonut viisi pelikenttää, sekä kenttäeditorin. Näille kaikille olen myös luonut erilaisia grafiikoita.

Pelissä hahmot voivat tehdä erilaisia liikkeitä vuoronsa aikana (Move, Light Attack, Heavy Attack, Break Mind Control, Rest ja Heal). Break Mind Control on erikoisliike, joka vaatii useita edellytyksiä toimiakseen: Kohteen on oltava heikentynyt (korkeintaan puolet elämästä jäljellä). Lisäksi liikkeen tekevän lemmikin on oltava kohteen viereisessä ruudussa, eikä se ole saanut tehdä mitään muita liikkeitä vuoronsa aikana, mukaan lukien siirtyminen ruudusta toiseen.

Ohjelmaa voidaan laajentaa muun muassa uusilla lemmikki-factoneilla ja uusilla tasoilla. Rakennan tekoälyn niin, että se osaa toimia sille annetuiden mahdollisuuksien mukaan, riippumatta siitä mitä ne ovat. Näin ollen tekoäly toimii myös jos sillä on ohjattavana uudenlaisia lemmikkejä joilla on uusia liikkeitä, sekä vastaavasti reagoimaan siihen, kun pelaajalla on pääsy uusiin lemmikkeihin ja liikkeisiin. Käyttöliittymän pitäisi toimia edelleen, jos laajennukset rakennetaan tottelemaan samoja sääntöjä, kuten aikaisempikin toiminnallisuus. Jos ei, vaaditaan pientä muokkausta.

Pelihahmojen grafiikat perustuvat 2d-spriteihin, sekä taustalla oleviin värillisiin kolmioihin, jotka osoittavat lemmikin tiimin. Pelikentät ovat taustakuvia, joiden päällä on esteitä, jotka nekin ovat spritejä.

Projekti on toteutettu vaativan vaikeustasun vaatimusten puitteissa.

3. Käyttöohje

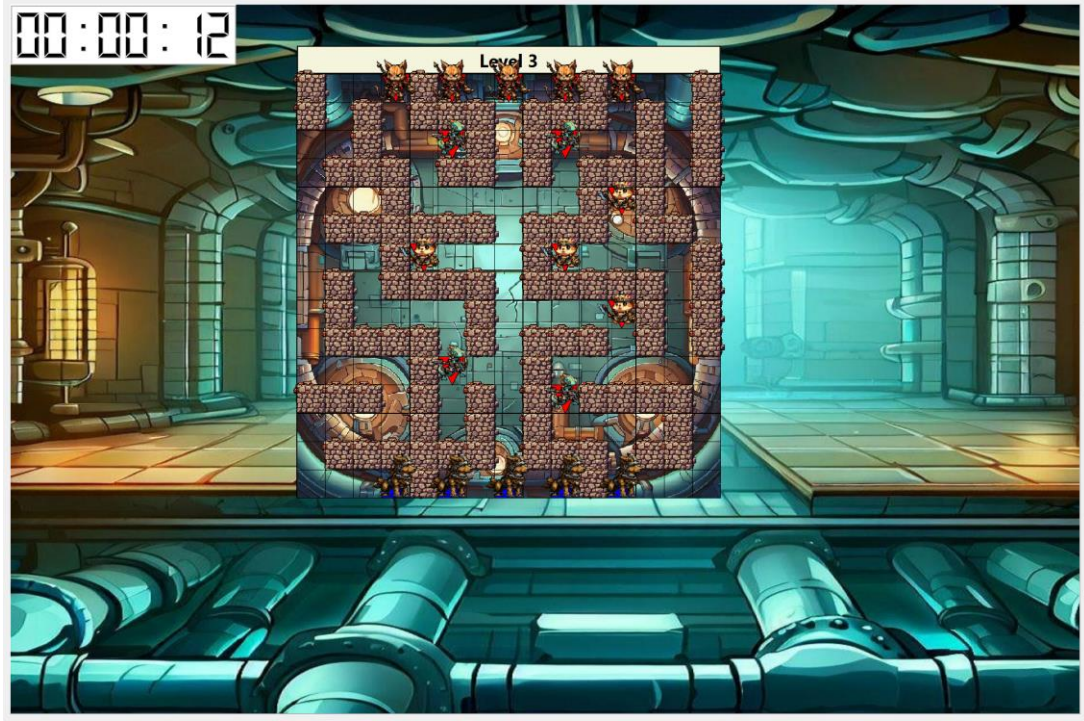
Kyseessä on videopeli, joten käyttöliittymä on graafinen. Peliä pelataan hiirtä käyttäen. Kenttäeditori vaatii näppäimistön käyttöä, kun halutaan tallentaa tasoja ja antaa niille nimi. Pelimaailma näkyy isometrisestä näkökulmasta, ja hahmot ovat eläimiä jotka liikkuvat ruuduilla, jotka korostuvat ja näyttävät mahdolliset liikkeet kun hahmoa halutaan liikuttaa. Kun haluat lopettaa vuorosi, paina ”End Turn” nappia. Tällöin tekoäly liikuttaa jokaista hahmoistaan älykkäästi, minkä jälkeen on taas sinun vuorosi.
Alla näytönkaappauksia tasoista ja kenttäeditorista:


00:01:08
Level 0
End Turn
Save and Quit
Load Game
00:00:05
Level 1
End Turn
Save and Quit
Load Game

End Turn

Save and Quit

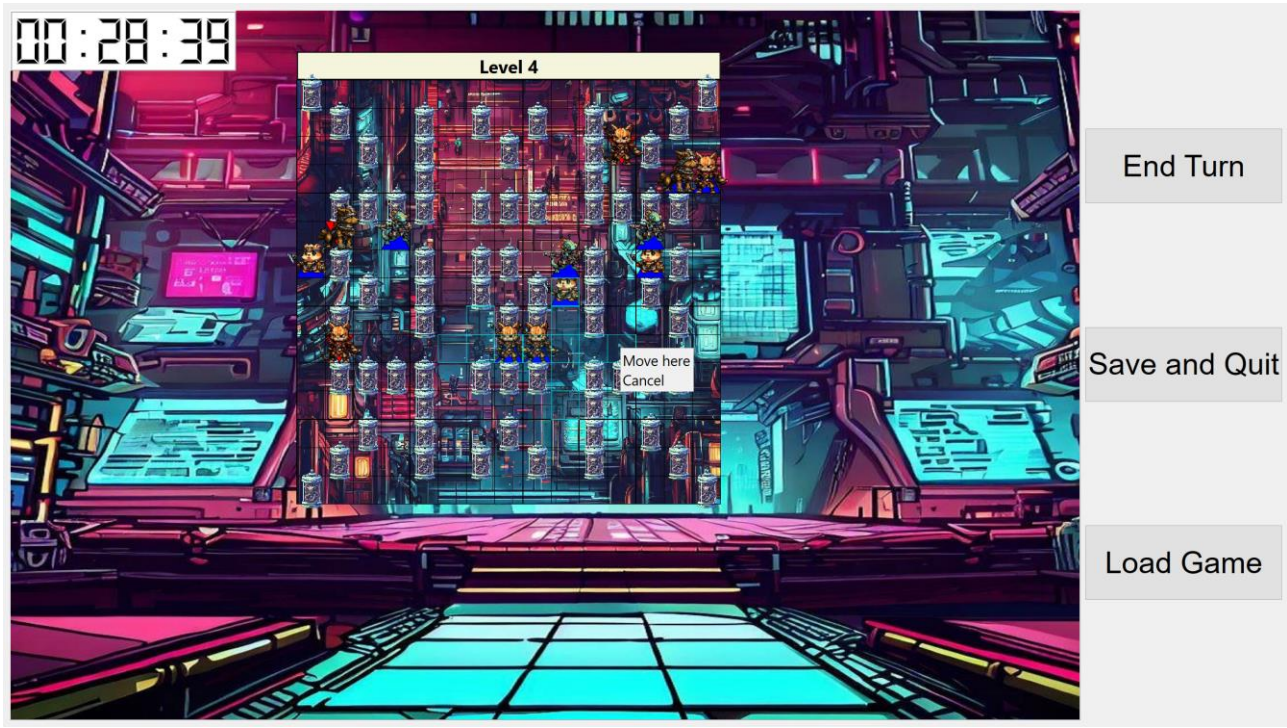
Load Game



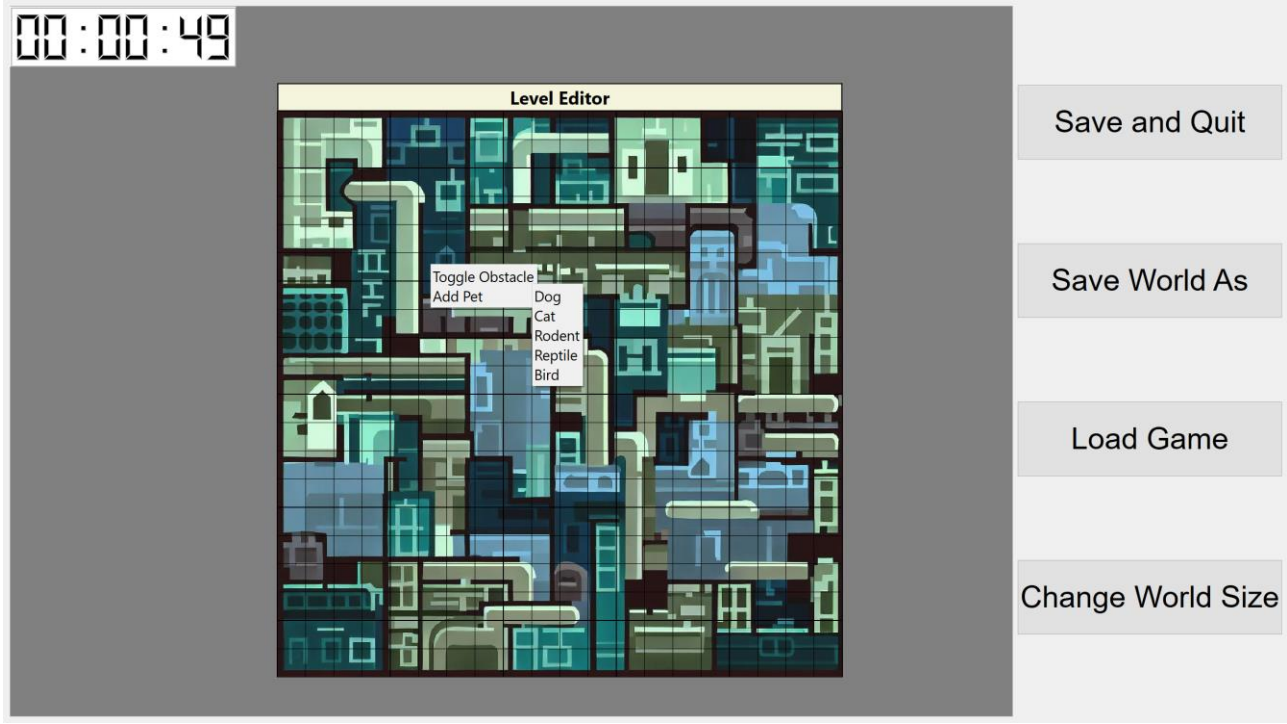
End Turn

Save and Quit

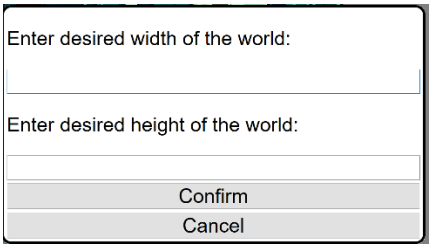
Load Game



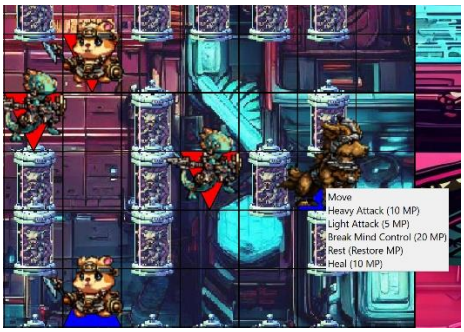
Kenttäeditorissa voidaan lisätä ja poistaa esteitä, sekä hahmoja klikkaamalla ruutua.



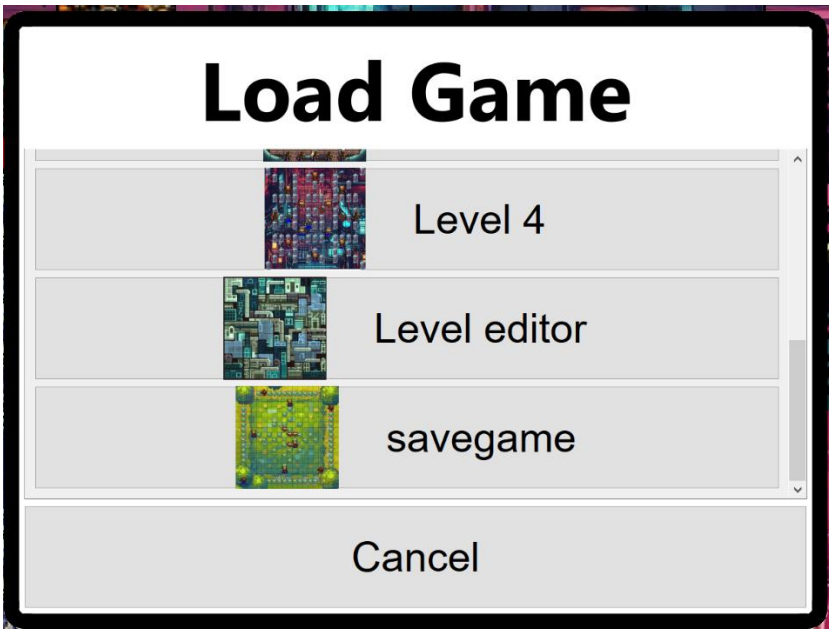
Kentän kokoa voidaan muuttaa ”Change World Size” painikkeen takaa:



Pelissä lemmikkiä voidaan ohjata klikkaamalla sitä ja valitsemalla liike avautuvasta valikosta:



Peli tallentuu jokaisen vuoron jälkeen automaattisesti savegame-tiedostoon. Tallentaessa peli ottaa näytönkaappauksen pelilaudan tilanteesta, joka näkyy tasovalikossa.



Esimerkkitilanne:

Kuvissa näkyy pelaajan vuoro. Pelaajan hahmo on kissa, jolla se hyökkää rottaan. Pelaaja yrittää murtaa aivopesun, josta rotta kärsii, vapauttaakseen tämän. Peli kuitenkin ilmoittaa pelaajalle, että tämä ei ole mahdollista, sillä vastustajaa on heikennettävä, ennen kuin vapauttaminen on mahdollista. Kun pelaaja on hyökännyt, on seuraavaksi tekoälyn vuoro ohjata rottaa. Taso on voitettu, kun kaikki viholliset ovat joko päihitettyjä tai ”vapautettuja”.

4. **Ulkoiset kirjastot**
   PyQt6 on ainut ulkoinen kirjasto, jota käytin.

5. **Ohjelman rakenne**
   Ohjelmassa on seuraavanlaisia luokkia.

   PetWorld: Lemmikkimaailma koostuu kaksiulotteisesta ruudukosta, jossa jokainen ruutu voi olla joko tyhjä, sisältää esteitä tai sisältää yhden lemmikin. Lemmikit voivat liikkua vapaasti tyhjillä ruuduilla, kunnes vastaan tulee toinen lemmikki tai este. Linnut kykenevät lentämään esteiden yli, tai laskeutumaan niille. Hyökätäkseen toiseen lemmikkiin, on lemmikin ensin siirryttävä tarpeeksi lähelle vihollista. Eri hahmojen hyökkäyksillä on eripituiset kantomatkat.

   Cat: cat-luokka on luokka lemmikkejä jotka ovat kissoja
   Dog: dog-luokka on luokka lemmikkejä jotka ovat koiria
   Rodent: rodent-luokka on luokka lemmikkejä jotka ovat jyrsijöitä
   Reptile: reptile-luokka on luokka lemmikkejä jotka ovat reptiilejä
   Bird: bird-luokka on luokka lemmikkejä jotka ovat lintuja

   AI: Luokka ohjaa vastustajan lemmikkejä käyttäen teko-älyä.

   GUI ja GuiExcercise: luokat, jotka määrittävät graafisen käyttöliittymän

   PetGraphicsItem: Luokka PetGraphicsItem laajentaa QGraphicsPolygonItem-luokkaa linkittääkseen sen lemmikin fyysiseen esitykseen. QGraphicsPolygonItem hoitaa piirtämisen, kun taas lemmikki tietää oman sijaintinsa ja tilansa. Lisäksi PetGraphicsItem on vastuussa

lemmikin kuvan piirtämisestä ja erilaisten lemmikkiin kohdistuvien toimintojen toteuttamisessa yhdessä lemmikkiluokan kanssa.

PetBrain: Luokka `PetBrain` edustaa kaksiulotteisissa ruudukkomaailmoissa asuvien virtuaalisten lemmikkien aivoja.

Square: Luokka Square edustaa yhtä neliötä lemmikkimaailmassa. Neliö voi sisältää joko esteen tai lemmikin tai se voi olla tyhjä.

Coordinates: Luokka Coordinates edustaa kokonaislukupareja, jotka määrittävät sijainnit kaksiulotteisessa ruudukossa. Kuten ohjelmointiympäristöissä on tavallista, x-arvot kasvavat oikealle (itään) päin ja y-arvot kasvavat alaspäin (etelään). Koordinaattiobjekti on muuttumaton luomisen jälkeen.

Direction: Luokka, jonka vakiot edustavat neljää pääilmansuuntaa, joissa lemmikit voivat liikkua.

6. Algoritmit

Olen luonut erilaisia algoritmeja pelin logiikan toteuttamiseksi. Yksi niistä on rekursiivinen algoritmi, joka antaa kaikki mahdolliset liikkumisvaihtoehdot lemmikille.

Funktio saa neljä parametria: x- ja y-koordinaatit, joiden avulla hahmon sijainti määritellään, r-parametri määrittelee hahmon liikkumisalueen säteen, ja esteet-parametri sisältää luettelon esteistä, jotka rajoittavat hahmon liikkumista. Jos parametrit jätetään tyhjiksi, ne saavat arvokseen funktiossa määritellyt oletusarvot.

Algoritmi alkaa tarkistamalla, onko hahmo kykenevä liikkumaan. Jos hahmon liikkumisalueen säde on nolla, se tarkoittaa, että hahmo ei voi liikkua ja algoritmi palauttaa hahmon nykyisen sijainnin.

Jos hahmo pystyy lentämään, algoritmi tuottaa kaikki mahdolliset siirtymät hahmon sijainnin perusteella. Se tutkii kaikki suunnat ja kaikki mahdolliset siirtymäetäisyydet, jotka ovat lyhyempiä tai yhtä suuria kuin liikkumisalueen säde. Lopuksi, algoritmi poistaa kaikki sijainnit, jotka ovat pelikentän ulkopuolella, ja lisää loput liikkumisvaihtoehtojen listaan.

Jos hahmo ei voi lentää, algoritmi tuottaa kaikki mahdolliset siirtymät kävellen. Tällöin algoritmi tarkistaa neljä suuntaa (ylös, alas, oikea, vasen) ja tarkistaa, onko uusi sijainti esteenä. Jos uusi sijainti ei ole este, algoritmi tutkii, onko se hahmon liikkumisalueen sisällä. Jos on, algoritmi löytää mahdolliset liikkumisvaihtoehdot uudelta sijainnilta, kutsuen itsensä rekursiivisesti ja lisää ne listaan. Lopuksi algoritmi

poistaa kaikki sijainnit, jotka ovat pelikentän ulkopuolella, ja lisää loput liikkumisvaihtoehtojen listaan.

Algoritmi poistaa lopuksi kaikki mahdolliset liikkeet, jotka ovat samoja kuin hahmon nykyinen sijainti, jotta vältetään turhat toistot. Lopuksi se palauttaa kaikki mahdolliset liikkeet liikkumisalueen sisällä.

Toinen toteuttamani algoritmi on vastustajan ”tekoäly”, joka määrää vastustajan lemmikkien liikkeet.

Algoritmi alkaa kutsuen aiemmin mainittua get_possible_moves-funktiota, joka palauttaa listan lemmikkieläimen mahdollisista siirroista pelilaudalla. Se tallentaa myös self.targets -nimiseen luokan muuttujaan kaikki pelaajan lemmikit.

Seuraavaksi funktio tarkistaa kaikki mahdolliset siirrot, jotka lemmikkieläin voi tehdä, ja tallentaa koordinaatit target-muuttujaan, jos mahdollinen siirto johtaa robotin paikalle, joka on tallennettu self.targets-muuttujaan. Jos lemmikkieläimen terveys on vähintään 50% ja sen mana on vähintään 5, se tarkistaa kaikki pelilaudalla olevat lemmikit world.robots-muuttujasta, tarkistaen, vastaako targets_dict-sanakirja lemmikin paikkaa target-muuttujassa. Jos ne vastaavat, attack-funktio kutsutaan lemmikin hyökkäämiseksi, minkä jälkeen lemmikkieläin liikkuu kohteen sattumanvaraiseen naapurikoordinaattiin. Jos tämä koordinaatti on päällekkäinen toisen lemmikin kanssa, siirrytään uuteen sattumanvaraiseen naapuriruutuun, kunnes vapaa ruutu löytyy.

Jos ei löydetä lemmikille kohdetta ja lemmikin mana on vähintään 10 ja sen terveys on alle täyden, lemmikki tekee sattumanvaraisen mahdollisen liikkeen ja sen terveys asetetaan täydeksi ja mana vähenee 10:lla. Jos taas sen mana on alle 10, lemmikin mana asetetaan täyteen, ja lemmikki ei liiku. Jos edellä mainitut ehdot eivät täyty, lemmikki tekee sattumanvaraisen mahdollisen liikkeen.

7. Tietorakenteet

Tehtävän monimutkaisuuden takia tulen luultavasti käyttämään monenlaisia tietorakenteita eri kokonaisuuksien toteuttamiseen. Pyrin pitkälti käyttämään Pythonin omia tietorakenteita, mutta tarvittaessa toteutan itse rakenteita. Esimerkiksi kaksinkerroin linkitetystä listasta (Doubly linked list) saattaisi olla hyötyä.

8. Tiedostot

Ohjelma käyttää valokuvatiedostoja pelien grafiikoiden toteuttamisessa. Näihin kuuluu esimerkiksi tasojen taustakuvat, hahmot ja erilaiset käyttöliittymän objektit. Lisäksi ohjelma kirjoittaa ja lukee tasojen tallennustiedostoja. Nämä tiedostot ovat

muotoa ”tasonimi.ptwrld”. Kyseessä on oma tiedostotyyppi, joka sisältää tavallista, ihmisluettavaa ja -muokattavaa tekstiä. Tiedostossa lukee tiedot tason ja lemmikkien statuksista, sekä viittaukset tason käyttämiin grafiikoihin. Lisäksi tiedostoon tallennetaan tason läpäisyn ennätysaika.

9. Testaussuunnitelma

Olen kokeillut pelin toiminnallisuutta pääasiassa ajamalla ohjelmaa ja pelaamalla peliä käyttöliittymässä. Lisäksi debugasin koodia print-komennoilla. Toteutin myös pari yksikkötestiä. Manipuloin myös pelin tallennustiedostoja erikoistilanteita varten. Olen myös toteuttanut apufunktioita ja widgettejä peliin, jotta erityistilanteita on helpompi testata. Esimerkiksi loin ”Debug” napin, jolle ohjelmoin erilaista toiminnallisuutta, kuten pelin ”lopettaminen” voittoon ajoissa. Lopullisesta ohjelmasta nämä metodit ovat kuitenkin poistettu. Esimerkkejä testatuista toiminnallisuuksista:

Pelin tallentaminen, sekä tallennuksen lataaminen. Onnistuuko kaikki hyvin, ilman että data korruptoituu? Onnistuuko pelaajan liikkeet odotetusti ja vaatimusten mukaisesti? Toimiiko tekoäly odotetulla tavalla? Onnistuuko lemmikin vapauttaminen?

10. Ohjelman tunnetut puutteet ja viat

Optimointi: Olen huomannut, että ohjelma alkaa hidastumaan pidemmän pelisession aikana. Tämä ongelma korjaantuu käynnistämällä ohjelma uudestaan. Koska peli tallentaa automaattisesti jokaisen vuoron jälkeen, ei tämä ole niin iso ongelma, ja koko pelin voi läpäistä käynnistämällä silloin tällöin ohjelma alusta. Lisäksi ohjelma pyörii hitaammin läppärillä kuin pöytäkoneella, jossa on tehokkaampi prosessori ja näytönohjain.

Jos jatkaisin projektia, niin keskittyisin koodin optimointiin. Käyttäisin tehokkaampia algoritmeja ja tietorakenteita, ja minimoisin näiden aikakompleksisuuksia. Lisäksi katsoisin mahdollisuuksia rajoittaa päivityksiä, sekä turhia ohjelmakutsuja.

Yritin myös toteuttaa aikaisemmassa vaiheessa animaatiota new record -leimalle, joka näkyy kun peli on voitettu ennätysajassa. Tarkoitus oli saada toteutettua eräänlainen leimausanimaatio. En kuitenkaan saanut tätä toimimaan.

Jos jatkaisin projektia, tutustuisin paremmin PyQt:n dokumentaatioon ja selvittäisin miksi animaatio ei toimi odotetulla tavalla.

11. 3 parasta ja 3 heikointa kohtaa

3 parasta:

1. Laajuus ja yksityiskohtaisuus: Olen toteuttanut enemmän luokkia ja toiminnallisuuksia kuin olisi tarvinnut tehtävänannon mukaan. Olen toteuttanut monia helppokäyttöisyys-ominaisuuksia, kuten zoomaaminen ja näytön liikuttaminen. Lisäksi elementit skaalautuvat eri näyttökokojen mukaan.
2. Visuaalisuus: Pelistä saa helposti kiinni sen intuitiivisen ulkoasun vuoksi. Esimerkiksi tasoa ladatessa, näkee näytönkaappauksesta heti, että miltä taso tulee näyttämään. Lisäksi tasoissa on hienot taustat ja grafiikat. Myös hahmot, sekä esteet ovat huolella suunnitellut.
3. Hiottu käyttökokemus: Olen itse pelannut pelin monta kertaa läpi, ja näin ollen löytänyt bugeja ja ongelmia koodissa ja korjannut ne hyvissä ajoin. Näin ollen peliä voi pelata pelkäämättä, että joutuisi kohtaamaan suuria ongelmia.

3 heikointa:

1. Aiemmin mainitsemani optimointi
2. Äänimaailma. Hyvissä peleissä on yleensä huolellisesti suunnitellut ääniefektit, sekä mukaansa tempaavat soundtrackit. Ohjelmassani ei ole ääntä tällä hetkellä ollenkaan, mutta mielellään säveltäisin kappaleita pelin tasoille jos sille löytyisi aikaa.
3. Animaatiot. Hahmoilla ei tällä hetkellä ole hyökkäys- tai idle-animaatioita. Ne olisivat hyvä olla.

12. Poikkeamat suunnitelmasta

Pysyin aika lailla suunnitelmassa ohjelman toteutuksessa. Päädyin kuitenkin lopulta priorisoimaan tiettyjä asioita eri tavoin kuin suunnitellusti. En esimerkiksi implementoinut peliohjain-tukea, mutta päädyin toteuttamaan näytönkaappaukset tiedostoihin ja mahdollisuuden muuttaa kentän kokoa suoraan kenttäeditorista, mitä en alun perin ollut suunnitellut.

13. Toteutunut työjärjestys ja aikataulu

Aikataulun suhteen en pysynyt niinkään suunnitellussa. Olin suunnitellut projektiin 100h työtä, eli noin 1.25 tuntia per päivä. Lopulta minulla saattoi hyvinkin mennä kaksinkertainen aika, sillä useampana päivänä käytin yli kymmenen tuntia projektiin, mutta tein projektia lähes joka päivä vähän.

Koska minulla aiempaa ohjelmointiprojektikokemusta ei ole, niin suunnittelu ei ollut hirveän helppoa konkreettisesti. Suunnitelmassani sanoin, että aloittaisin luokkien ja pelilogiikan rungoista ja vasta myöhemmin alkaisin tekemään käyttöliittymää. Lisäksi

sanoin, että vasta paljon myöhäisemmässä vaiheessa hiirtä voitaisiin käyttää pelin ohjaamiseen.

Todellisuudessa käyttöliittymää ja pelilogiikkaa kehitin käsi kädessä samaan tahtiin, ja hiiri toimi alusta lähtien ohjaustyökaluna.

14. Arvio lopputuloksesta

Kaiken kaikkiaan olen hyvin tyytyväinen projektini lopputulokseen. Minusta tuntuu, että olen oppinut valtavasti sitä tehdessä, ja uskon, että intohimoni projektia kohtaan paistaa läpi sitä pelatessa ja arvioitaessa.

Ohjelman toiminnallisuus on erinomainen, ja se täyttää sille asetetut vaatimukset moitteettomasti. Käyttöliittymä on intuitiivinen ja helppokäyttöinen, mikä helpottaa pelin oppimista ja pelaamista ja tarjoaa miellyttävän käyttäjäkokemuksen.

Ohjelman suorituskyky on hyvä, tosin hidastuu kun peliä pelataan tarpeeksi pitkään.

Luotettavuus on toinen ohjelman vahvuuksista. Se on varustettu vahvalla virheidenkäsittelyllä ja poikkeustenhallinnalla, mikä tekee siitä turvallisen ja vakaan käyttää. Ohjelma osaa käsitellä erilaisia virhe- ja poikkeustilanteita tehokkaasti, ja sen ei pitäisi kaatua sitä pelatessa.

Ylläpidettävyyden kannalta ohjelman rakenne on erinomainen. Lähdekoodi on hyvin dokumentoitu ja siinä on selkeät kommentit, mikä helpottaa koodin ymmärtämistä ja muokkaamista. Testaamisen ja virheenjäljityksen työkaluja on hyödynnetty tehokkaasti, mikä helpottaa ohjelman ylläpitoa ja kehittämistä.

Ohjelman laajennettavuus on myös huomattava vahvuus. Sen rakenne on suunniteltu tulevia muutoksia ja laajennuksia varten, mikä mahdollistaa joustavan sopeutumisen uusiin vaatimuksiin. Ohjelman rakenne tukee muutoksia ja laajennuksia, mikä tekee siitä erinomaisen valinnan pitkäaikaiseen kehitystyöhön. Esimerkiksi uusia lemmikkityyppejä ja hyökkäyksiä ja ominaisuuksia on helppo lisätä.

Kaiken kaikkiaan ohjelman toteutus on erinomainen. Sen vahvuuksina korostuvat hyvä toiminnallisuus, suorituskyky, luotettavuus, ylläpidettävyys ja laajennettavuus. Ohjelma tarjoaa käyttäjilleen erinomaisen kokemuksen ja ennen kaikkea sitä on hauska pelata.

15. Viitteet

Kurssimateriaalit CS-A1121 Ohjelmoinnin peruskurssi Y2 (2023) *CS-A1121 Ohjelmoinnin peruskurssi Y2*. Available at: https://plus.cs.aalto.fi/y2/2023/ (Accessed: 12 May 2023).

(2023) *Python 3.11.3 documentation*. Available at: https://docs.python.org/3/ (Accessed: 12 May 2023).

PyQt Documentation v6.5.0 (2023) *Reference Guide - PyQt Documentation v6.5.0*. Available at: https://www.riverbankcomputing.com/static/Docs/PyQt6/index.html (Accessed: 12 May 2023).

(2023) *Python Wiki*. Available at: https://wiki.python.org/moin/ (Accessed: 12 May 2023).

The hitchhiker's guide to python! - the hitchhiker's guide to python (2011). Available at: https://docs.python-guide.org/ (Accessed: 12 May 2023).