1. **Henkilötiedot**

   Otsikko: Pet Wars: Rise of Fluffy
   Tekijän nimi: Rasmus Kull
   Opiskelijanumero: 884433
   Koulutusohjelma: CS-A1121
   Vuosikurssi: 2023
   Päiväys: 20.02.2023

2. **Yleiskuvaus ja vaikeustaso**
   Strategiapeli

   Teen vuoropohjaisen strategiapelin tekoälykkäällä tietokonevastustajalla. Käytän inspiraationa XCOM -pelisarjaa. Käytän 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 hyökkäyksen kohteensa laskukaavan perusteella. Laskukaavaan vaikuttaa vihollisen etäisyys, vihollisen jäljellä oleva elämä (HP), sekä eläimien väliset vahvuuskertoimet (esim. koirat ovat vahvoja kissoja vastaan, kissat lintuja jne.). Hyökkäys vastaavasti valitaan kohteen jälkeen kaavalla, johon vaikuttaa hyökkäyksen vahvuus vihollista vastaan, jäljellä olevat hyökkäyspisteet (PP), sekä itsesuojelu (esim. jos mahdollista/tarpeellista, käyttää healaus-kykyä).

   Pelinsynopsis 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, ja eri lemmikki-factioneiden sisällä on erilaisia eläimiä, jotka edelleen erottuvat toisistaan. Lemmikeillä on myös erilaisia hyökkäyksiä. Pelissä on erilaisia tasoja ja ympäristöjä, joissa joudut taistelemaan eri factioneitä vastaan. Näissä ympäristöissä on erilaista maastoa, joka vaikuttaa eri eläimiin eri tavalla. Esimerkiksi kissat inhoavat vettä, joten he eivät voi ylittää

lammikkoa. Jyrsijät kykenevät liikkumaan pienistä koloista, mistä isommat eläimet eivät pääse läpi jne.

Different pet factions: Cats, Dogs, Rodents, Reptiles and Birds

Koiratasot ovat esikaupunkialueella, taustalla omakotitaloja, Kissoja on kaupungeissa, rottia viemäreissä ja reptiilejä viidakossa. Linnut viihtyvät kaikkialla.

Pelissä on useita statusvaikutuksia, kuten bleed, poison, burning, jotka heikentävät eläimiä. Tietyt eläimet myös kykenevät valtaamaan toisia eläimiä väliaikaisesti aivopesulla. Myös positiivisia vaikutuksia on, jotka voimaannuttavat eläimiä.

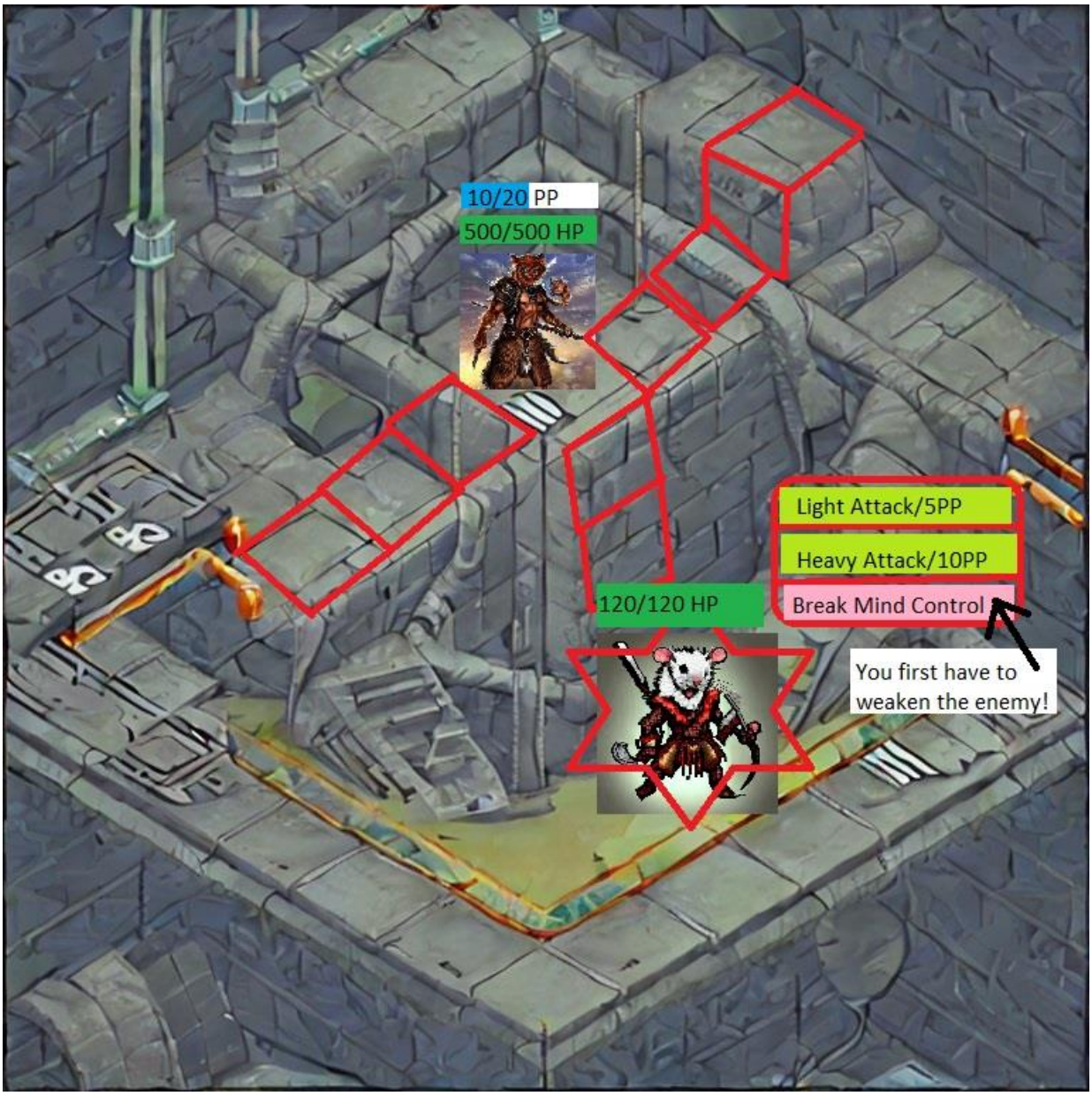
Ohjelmaa voidaan laajentaa vaikka uusilla lemmikki-factoneilla ja uusilla tarinoilla ja tasoilla. Rakennan tekoälyn niin, että se osaa toimia sille annetuiden mahdollisuuksien mukaan, riippumatta siitä mitä ne ovat. Näinollen tekoäly toimii myös jos sillä on ohjattavana uudenlaisia lemmikkejä joilla 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, joiden animaatiot ovat joko hyvin yksinkertaisia tai olemattomia. Pelikentät ovat taustakuvia, joiden päällä on objekteja jotka vaikuttavat hahmoihin eri tavoilla.

Tähtään vaikeustasoksi tasoa vaikea.

3. Käyttötapauskuvaus ja käyttöliittymän luonnos

Kyseessä on videopeli, joten käyttöliittymä on graafinen. Peliä pelataan hiirellä klikaten, tai vaihtoehtoisesti peliohjaimella pelatessa valitsemalla objekteja joystickillä ja painamalla nappeja. Pelimaailma näkyy isometrisestä näkökulmasta, ja hahmot ovat eläimiä jotka liikkuvat virtuaalisilla ruuduilla, jotka tulevat näkyviin kun hahmoa painetaan.
Alla konseptikuva siitä, miltä viemäri-taso voisi mahdollisesti näyttää:



Kuvassa 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. Ohjelman rakennesuunnitelma

Ohjelmassa on seuraavanlaisia luokkia.

PetWorld: Lemmikkimaailma koostuu kaksiulotteisesta ruudukosta, jossa jokainen ruutu voi olla joko tyhjä, sisältää erilaisia esteitä tai sisältää yhden lemmikin. Lemmikit voivat liikkua vapaasti tyhjillä ruuduilla, kunnes vastaan tulee toinen lemmikki tai este. Osa lemmikeistä kykenee liikkumaan tiettyjen esteiden yli, tai käyttämään vuoronsa esteen tuhoamiseen.

Hyökätäkseen toiseen lemmikkiin, on lemmikin ensin siirryttävä tarpeeksi lähelle vihollista. Eri hyökkäyksillä on eripituiset kantomatkat, mutta useimmat hyökkäykset vaativat kohteen olevan viereisessä ruudussa.

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

Jokaisella luokalla on omat ala-luokkansa lemmikeistä. Esimerkiksi Rrodent-luokan alaluokkia ovat rat, mouse, hamster, gerbil ja guinea_pig

GUI: luokka, joka määrittää graafisen käyttöliittymän

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

PetBrain: Luokka `PetBrain` edustaa kaksiulotteisissa ruudukkomaailmoissa asuvien virtuaalisten lemmikkien "aivoja" (tai tekoälyä, AI). Lemmikkieläimen aivoissa on algoritmi, joka määrittää, mitä lemmikin pitäisi tehdä pelivuoronsa aikana. Toisin sanoen lemmikkieläimen aivot kykenevät ohjaamaan lemmikkieläimen kehon toimia.

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.

5. **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.

6. **Tiedostot ja tiedostoformaatit**

Ohjelma käyttää valokuvatiedostoja pelien grafiikoiden toteuttamisessa. Näihin kuuluu esimerkiksi tasojen taustakuvat, hahmot ja erilaiset käyttöliittymän objektit. Lisäksi ohjelma kirjoittaa ja lukee pelaajan tallennustiedostoja. Nämä tiedostot ovat muotoa ”pelaaja1.ptwrld”. Kyseessä on oma tiedostotyyppi, joka sisältää tavallista tekstiä. Tiedostossa lukee pelaajan hahmojen ominaisuudet ja tieto pelaajan etenemisestä (esim. mitkä tasot on läpäisty ja kuinka nopeasti).

7. Algoritmit

Tekoälyn algoritmi on jokseenkin seuraavanlainen: Tekoäly pisteyttää kaikki mahdolliset liikkeet, ja valitsee lopuksi liikkeensä sen perusteella, millä on eniten pisteitä. Liikkeitä on seuraavanlaisia:

- Heal(self)
- Heal(other)
- Light Attack
- Heavy Attack
- Rest

Heal(self) laskee menetetyn elämänsä osuuden prosentteina. Jos esimerkiksi hahmon maksimi HP on 100 ja hahmolla on enää 66hp jäljellä, tulee pisteiksi 0.44

Heal(other) toimii samalla periaatteella, ja se lasketaan jokaiselle tekoälyn ohjaamalle muille hahmoille erikseen.

Jos Heal(self) = Heal(other), niin tekoäly valitsee aina Heal(self).

Jos useampi Heal(other) saavat saman arvon, valitsee tekoäly kohteen satunnaisesti.

Light Attackin pisteytyskaava on seuraavanlainen: damage inflicted as % of max health * 1.5 + 1 if results in KO

Heavy Attackin pisteytyskaava on seuraavanlainen: damage inflicted as % of max health + 1 if results in KO

Rest palauttaa hahmon PP:t täyteen arvoon. Hahmot saavat myös 5 PP:tä vuoronsa alussa. Restin pisteytyskaava on seuraavanlainen: PP recovered * 0.01

Light ja Heavy Attack lasketaan erikseen jokaiseen mahdolliseen kohteeseen. Jos pistemäärä on sama useammassa kohteessa, valitsee tekoäly satunnaisesti kohteensa.

8. Testaussuunnitelma

Pelin tallentaminen, sekä tallennuksen lataamista on testattava. Onnistuuko kaikki hyvin, ilman että data korruptoituu? Onnistuuko pelaajan liikkeet odotetusti ja vaatimusten mukaisesti? Toimiiko tekoäly odotetulla tavalla? Onnistuuko lemmikin vapauttaminen? Voiko peliä ohjata sekä ohjaimella, että hiirellä? Näitä kaikkia voidaan kokeilla pelaamalla peliä. Erityistilanteita voi kokeilla manipuloimalla pelin tallennustiedostoa tekstinkäsittelyohjelmalla ennen tallennuksen lataamista.

9. Kirjastot ja muut työkalut

En tunne mitään pythonin kirjastoja, joten käytän lähtökohtaisesti vain pythonin peruskirjastoja, jos niiden käyttö on sallittua. Jos osa pelini toiminnoista vaatii muita kirjastoja niiden toteuttamiseen, tiedustelen asiaa kurssiassistentiltä.

10. Aikataulu

Projektin laajuus on 3 opintopistettä, koska se on 60% kurssiarvosanasta. 3op = 27*3 = 81h. Koska minulla ei ole aiempaa kokemusta pelien kehityksestä, arvioin, että minulla tulee menemään ainakin 100h projektin tekemiseen. Kurssin deadlineen on 80 päivää. Ensimmäiseen välipalautukseen on 31 päivää. Toiseen välipalautukseen on 52 päivää. 100/80 = 1.25. Tulen siis käyttämään keskimäärin joka päivä 75 minuuttia projektiin.

Ensimmäiseen välipalautukseen mennessä aion toteuttaa tärkeimpien luokkien rungot ja perustoiminnallisuudet. Seuraan jo alustavasti mahdollisten bugien syntymistä ja korjailen niitä samalla. Alan myös tekemään käyttöliittymää, en kuitenkaan keskity siihen vielä sen kummemmin. Työmäärä tunneissa: 31 * 1.25 = 38.75h

Toiseen palautuskertaan mennessä tähtään siihen, että kaikki luokat ja koodi toimii hyväksyttävästi niin, että peliä on jo mahdollista pelata. Tähtään siihen, että käyttöliittymä toimii, ja että sekä hiirtä ja ohjainta voidaan jo käyttää pelin ohjaamiseen, ja että tallennukset ja niiden lataaminen toimii. Työmäärä tunneissa: 21 * 1.25 = 26.25h

Toisen välipalautuksen ja lopullisen palautuksen välillä keskityn pelikokemuksen hiomiseen. Tähän kuuluu graafisen puolen lopullinen toteutus, eli pelikenttien ja hahmojen ulkonääöt, sekä pelin testaaminen eri tilanteissa ja bugien korjaaminen. Työmäärä tunneissa: 28 * 1.25 = 35h

11. Kirjallisuusviitteet ja linkit

Tulen käyttämään tämän kurssin kurssimateriaaleja, PyQt6-käyttöliittymäkirjastoa. Muita resursseja joita saatan hyödyntää ovat Python wiki, The Hitchhiker's Guide to Python! Ja mahdollisesti tiettyihin ongelmin StackOverflowta.

Löysin myös kiinnostavan sivun netistä: 2D Strategy Game (https://www.patternsgameprog.com/series/2d-strategy-game/). Siinä kuvataan 2D

strategiapelin tekemistä Pythonilla. Tiedostan, että kyseinen sivu käyttää kiellettyjä kirjastoja, mutta sivusta voi silti olla hyötyä esimerkiksi inspiraationm lähteenä.

12. Liitteet

Mahdollisia liitteitä. En tiedä saako/tarviiko niitä olla, mutta jos niille on tarvetta liitän ne projektiin.