1. **Henkilötiedot**

   StrategiaPeli; Lauri Paloneva, 908445, Kauppatieteiden kandidaatti, 12.05.2023.

2. **Yleiskuvaus**

   Vuoropohjainen strategiapeli, jossa pelaajalla, sekä tekoäly vastustajalla on 4x4 kokoinen ruudukko pelialustana. Molemmilla osapuolilla on ruudukolla 1–3 hahmoa. Vuorolla pelaaja voi päättää haluaako hän tehdä iskun vai liikuttaa jotain hahmoista. Vuorollaan voi ainoastaan tehdä yhden iskun tai yhden siirron. Pelin tavoitteena on tuhota vastustajan hahmot, ennen kuin vastustaja tuhoaa omat hahmot. Iskut tekevät eri määrän vahinkoa hahmoihin, sekä osuvat eri alueelle 4x4 ruudukkoa. Esimerkiksi yksi isku voi tehdä 2 elämäpistettä vahinkoa ja vaikuttaa 2x2 kokoiselle alueelle. Toinen isku voi tehdä 5 elämäpistettä vahinkoa, mutta vaikuttaa vain yhteen ruutuun. Hahmoilla on nopeuselementti, joka määrittää kumman pelaajan isku tapahtuu ensin. Tämän lisäksi hahmoilla on eri elämäpisteitä, jotka määrittävät montako iskua hahmo kestää. Jos pelaaja siirtää hahmoa tämä tapahtuu aina ennen iskua, vaikka hahmo olisikin hitaampi. Hahmoihin voi kohdistua erilaisia vaikutuksia. Näitä voivat aiheuttaa Isku tai pelialustan laatta. Esimerkiksi Myrkkylaatta aiheuttaa hahmolle myrkytyksen, joka aiheuttaa vahinkoa vuoron jälkeen. Muita vaikutuksia on esimerkiksi liikkumisen estyminen. Grafiikkana toimii 2 kpl 4x4 pelialustoja. Hahmot ovat erilaisia muotoja, kuten ympyröitä ja kolmioita. Tekoäly vastustaja noudattaa strategiaa, jossa se pyrkii iskemään eniten kokonaisvahinkoa tekevän iskun tai siirtämään vähä elämäpisteisen pelaajan turvaan. Alustavasti olen ajatellut suorittaa projektin vaativana versiona.

3. **Käyttöohje**

   Tämän voi osittain käydä läpi README-tiedostossa. Älä kuitenkaan toista tässä samoja asioita kuin README-tiedostossa.

   Ohjelma käynnistetään ajamassa GUI tiedosto. Tämän jälkeen valitaan ”valitse tiedosto” napista, Dokumentit kirjastossa oleva ”Config_file”.

   Tämä käynnistää pelin. Nyt avautuu näkymä, jossa on Start Game, Add Player, Load Game Napit, sekä Select player ja select Tile Type napit. Klikkaamalla alempaa ruudukkoa, voi lisätä omalle kentälleen hahmon. Valitsemalla select player valikosta eri nimen, voi lisätä eri hahmon. Kun haluaa lisätä erilaisen laatta tyypin, voi klikata add a player napista, jolloin teksti muuttuu ”Add a Tile”:ksi. Tämän jälkeen klikkaamalla oman kentän ruutua, laatta muuttuu valituksi Tileksi. Itse laittamat Tilet peilautuvat vastustajan kenttään.

   Jos haluat uudelleen sijoittaa jonkun hahmoista, valitse ”Add a player” näkyville, ja klikkaa kentällä olevaa hahmoa, niin se poistuu. Nyt voit lisätä uuden hahmon. Kun kentällä on 3 hahmoa, et voi lisätä enään hahmoja. Kun olet tyytyväinen kenttään, voit valita Start Game, joka aloittaa pelin ja sijoittaa vastustajan hahmot.

   Jos olet aiemmin tallentanut pelin, voit valita Load Game, jolloin voit valita tiedostoistasi tallennus tiedoston, johon olet tallentanut pelin tiedot. Tätä varten on dokumentit osiossa save_file.

   Kun peli on alkanut, on näkymässä vastustajan pelaajien infot ylimpänä.

Jos haluat hyökätä hahmolla, pidä huoli, että ”Selecting Player for Attack” Teksti on aktiivinen, Yksittäisen playerInfon yläpuolella. Jos haluat liikuttaa pelaajaa, valitse Move Player, jolloin tekstiksi muuttuu ”Moving Player”.

Kun olet valinnut päätöksesi, klikkaa pelaajaa, jolla haluaisit tehdä liikkeen. Jos haluat hyökätä, voit pelaajan valittuasi, klikata vastustajan puolen ruudukkoa, johon ilmestyy punaisella rajattu alue. Jos klikkaat aluetta, jossa vastustajan pelaaja on, huomioi, että valitaksesi hyökkäysalueen, sinun tulee klikata ruudun, aluetta, jossa pelaaja ei ole. Muuten valitset pelaajan tarkasteltavaksi PlayerInfo kohtaan. Jos klikkaat vahingossa vastustajan hahmoa, klikkaa tämän jälkeen uudestaan sitä hahmoa, jolla haluat hyökätä.

Kun olet tyytyväinen alueeseen, klikkaa ”ATTACK”-nappia. Tämä toteuttaa iskun. Vaststajan tekemän iskun alue, näkyy keltiasella omassa ruudussasi.

Vuoron tapahduttua, Klikkaa ”Next Turn” Painiketta, niin että tekstiksi muuttuu ”Turn Active”. Tämän jälkeen voit jatkaa seuraavaan vuoroon.

Jos haluat liikkua, valitse atkiiviseksi ”Moving Player”. Kun nyt klikkaat hahmoa, värittyy osa kentästä vihreillä reunoilla. Valitse joku näistä laatoista klikkaamalla sitä. Tämän jälkeen laatta värittyy punaisilla reunoilla. Nyt kun klikkaat MAKE MOVE. pelaajaa siirtyy.

Painamalla SAVE napista, voit valita, mihin tiedostoon tallennat pelin tilanteen.

4. **Ulkoiset kirjastot**

   Ei ulkoisia kirjastoja

5. **Ohjelman rakenne**

   Ohjelman erottelu tärkeimpiin osakokonaisuuksiinsa, toteutuneen luokkajaon esittely. Minkälaisilla luokilla kuvaatte ohjelman ongelma-aluetta? Mitä ongelman osaa kukin luokka mallintaa? Mitkä ovat luokkien väliset suhteet? Entä millaisia luokkia käytät ohjelman käyttöliittymän kuvaamiseen? Valmiita Python-luokkia ei tarvitse esitellä samalla tavalla kuin omia, mutta tehdyt valinnat on hyvä perustella.

   Tässä voi esittää myös mahdollisia muita ratkaisumalleja ja näin perustella valittu ratkaisu. Jos suinkin mahdollista, liitä mukaan jonkinlainen graafinen luokkakaavio (voit käyttää esim. UML-luokkakaavionotaatiota, mutta se ei ole millään muotoa pakollista). Esittele omien luokkien keskeiset metodit. Huom. oleellista on vain se, mitä metodeilla tehdään, ei se, miten ne sisäisesti toimivat.

   Ohjelman pääKomponentti on GUI luokka, jonka varaan suurin osa toiminnallisuudesta rakentuu.

   GUI luokkaan importataan Gamegrid, tileGraphicsItem, Player, playerGraphicsItem, menuButtons, GameButtons, Location, ReadFile, Turn, sekä ai luokat.

Gui luokan keskeiset metodit ovat GameBegin ja startGame, jotka luovat pelin ja aloittavat sen.

Menu buttons, luo grafiikkaan kenttäeditorin napit, ja GameButtons luo pelin aikaiset napit.

MenuButtonsissa on __init__ metodin lisäksi, set visible metodi, joka määrittää näkyvätkö napit vai eivät.

GameButtonssissa on useita metodeita jotka päivittävät pelaajien infoja tekstikenttiin. attack_or_move metodi, määrittää, napin tekstin, jolla päätetään hyökätäänkö vai liikutaanko.

Ohjelmassa on Gamegrid, joka luo gridit ja player luokka joka luo pelaajien toiminnallisuuden. Gamegridin tärkeimmä metodit ovat, get_players jolla haetaan kaikki Gamegridn pelaajat, sekä get_tile, jolla haetaan tietyn koordinaatin tile.

player luokkaan importataan Location luokka.

player luokan tärkeimpiä metodeita ovat get_move_area(), jolla haetaan alue, jolle pelaaja voi liikkua, sekä is_hit, jolla toteutetaan iskun vaikutus.

playerGraphicsitem ja tileGraphicsItem luovat pelaajien ja laattojen grafiikat.

Näiden tärkeimmät metodit ovat mousePressEvent, joka totetuttaa painalluksesta tapahtuvat toiminnallisuudet.

playerGraphicsitemiin importataan Location ja Attack luokat.

tileGraphicsItemiin importataan Attack luokka.

read_file lukee Konfigurointi tiedoston. Sen tärkein luokka on read_info_file, jolla luetaan Konfigurointi tiedosto.

Attack luokka luo hyökkäysten toiminnallisuuden. Siihen importataan Location luokka. Sen tärkein luokka on get_hit_area, jolla saadaan iskun alue.

Effect luokka on keskeneräinen, eikä sen toiminnallisuutta käytetä.

Turn luokkaa määrittää, kumpi pelaaja iskee ensin. sen ainut metodi on moves_first, joka määrittää kumpi pelaaja iskee ensin.

Location luokka, tallentaa locaation ja sen kautta voi palauttaa koordinaatit. sillä on kaksi luokkaa get_x j get_y jolla palautetaan koordinaattien arvot.

ai luokka luo vastustajan tekoälyn toiminnallisuudet. Siihen importataan Location ja Attack luokat. Sillä on decide_to_move, decide_best_move_point sekä decide_best_attack luokat. Näillä määritetään mikäli kannattaa liikkua, paras liikkumispiste, sekä paras hyökkäys ja sen iskupiste.

6. **Algoritmit**

Pelin tekoäly valitsee parhaimman iskun, pelaajien iskujen kokonaisvaikutuksen perusteella. Algoritmi looppaa läpi kaikki pelaajat ja näiden iskut, jokaiselle iskulle algoritmi käy läpi kaikki mahdolliset iskupaikat. Jokaiselle mahdolliselle iskupaikalle lasketaan score aluella olevien vastustajan pelaajien perusteella. scoreen lisätään iskun kokonaismäärä. Jos jokin pelaaja tuhoutuisi iskusta, lisätään scoreen tällöin kolminkertainen määrä iskun vaikutuksesta.

Algoritmi palauttaa parhaan laatan, johon iskeä, parhaan pelaajan grafiikka hahmon, sekä parhaan iskun.

Tekoälyllä on myös algoritmi, joka valitsee parhaan liikkumis kohdan. algoritmi käy läpi laatat pelaajan mahdollisella liikkumisalueella. tämän jälkeen, katsotaan, onko laatan tyyppi parempi, kuin edellisen. Jos Tile ei ole tyhjä, miinustetaan scoresta 1000, jotta ei liikuttaisi jo varattuun laattaan. Scoreen listään laatta tyyppien erotus kerrottuna viidellä, sekä sen koordinaattien etäisyyden itseisarvo. pelaaja pyrkii siis liikkumaan mahdollisimman pitkälle.

Kolmas algoritmi päättää, liikkuuko vastustaja vai ei. käy läpi pelaajat, ja tarkistaa, onko niiden elämäpisteet alle, kriittisen rajan. Jos on, se tarkistaa kellä pelaajista on matalimmat elämäpisteet ja palauttaa tämän pelaajan. jos kaikkien pelaajien pisteet ovat yli rajan, palautetaan 0.

Myös python random algoritmia käytetään.

7. **Tietorakenteet**

   Tietorakenteina käytettiin listoja, ja sanakirjoja. Konfigurointi tiedostosta luetaan pääasiassa sanakirjoihin, joista tieto luetaan eri luokkiin. Pelaajien määrät, eri laatat yms on tallennettu listoihin. Käytin Muuttuva tilaisia rakenteita, jotta esim pelaajien määrä yms voidaan muokata. Listat soveltuivat parhaiten, koska niiden käyttö on yksinkertaista.

8. **Tiedostot**

   Ohjelmassa on python tiedostojen ohella Konfigurointi tiedosto ”Config_file” sekä tallennus tiedosto, ”save_file”, sekä testiä varten ”temp_file”, jonka se lukee testaamista varten. ”temp_file” on koodit osiossa, sillä en kerennyt muokata testiä niin, että tiedoston voi valita.

   Tiedostot ovat tekstitiedostoja. Config_file tiedostossa on lueteltu eri pelin osaalueet, sekä niihin liittyvät tiedot. Ohjelma tarvitsee Config_file:n sellaisenaan, jotta se toimii.

   Eri osa-alueiden data on otsikoitu ja väliotsikoita. otsikko on isoilla kirjaimilla ja väliotsikot sisältävät”:” lopussa. NEWBLOCK teksti aloittaa uuden otsikon.

   STOP käsky lopettaa tiedoston luvun.

Config filessä voi eri numeroita muutamalla muokata iskujen voimaa, vaikutusta, hahmojen, hahmojen maksimi määrää yms. Myös pelaajien nimiä voi muokata. Tiedosto tyypissä teksti data pitää erottaa pilkulla ja kirjoittaa yhteen.

Save_fileen on tarkoitus tallentaa pelin tiedot. Kun klikataan Save, peli tallentaa tiedot sellaisessa formaatissa, että kun sen lataa ”load_game” kohdassa, pelin ymmärtää sen formaatin.

9. **Testaus**

   Ohjelma sisältää minimaalisen määrän testejä, sillä aika loppui kesken näiden tekoon. Ohjelmaa tehdessä, sitä testattiin pääasiassa debuggerilla, sekä printtaamalla arvoja tietyissä väleissä.

   Ohjelma läpäisee Move_player testin, joka sille on tehty. Ohjelman Testaaminen jäi hyvin brute force tasolle, pelaamisen kautta, sillä ohjelman kehittäminen vei niin paljon aikaa, ja se loppui kesken.

   Testilukee tempfilen, jossa on pelin konfigurointi tiedot.

10. **Ohjelman tunnetut puutteet ja viat**

    Ohjelma testaus on mitätöntä. Koodin siisteysarvo on heitetty lähes roskakoriin ajan kanssa painimisen takia. try, except lauseet ja errorien nostaminen, sekä koodin kommentointi puuttu kokonaan.

    Ohjelma saattaa olla herkkä joidenkin nappi yhdistelmien väärälle järjestykselle.

    Tekoälyn liikkumisominaisuus saattaa jumittua muutaman liikeen jälkeen, niin, että vastustaja liikkuu, mutta päättääkin että paras paikka on se missä nyt ollaan.

    Efektien vaikutusten pitäisi enimmäkseen toimia, mutta niiden lisääminen hahmoille on hieman satunnaista.

    Jos saisin jatkaa, hioisin koodin siistiin kuntoon, ja lisäisin testejä. Lisäksi lähtisin tarkistamaan kohtia, missä ohjelma saattaa kaatua, ja tekisin näille errorien nostoja sopiviin kohtiin. Isoin haaste Projektissa oli sen laajuus, ja osa ominaisuuksista jäi ihan alkutekijöihin

11. **3 parasta ja 3 heikointa kohtaa**

    Ohjelman parhaat puolet, ovat pelattavuudessa. Peli näyttää miellyttävältä, iskut toimivat ja tekoäly tekee oikeita isku valintoja. Pelin pystyy pelata alusta loppuun useissa tapauksissa.

    Ohjelman heikkoudet ovat ehdottomasti koodin siisteydessä, sekä testauksissa ja kaatumisen ennakoimisessa. Ohjelman ei pitäisi kovin usein kaatua, mutta se ei sisällä minkään laisia varatoimia tätä vastaan.

12. **Poikkeamat suunnitelmasta**

Poikkesin jonkin verran suunnitelmasta. Esimerkiksi pelaajista ja laatoista en tehnyt periytymis- luokkia, sillä ominaisuudet oli helppo luoda muuttujien avulla. Ajankäytön suunnittelu meni ihan pieleen. Projektiin meni noin 3-4 kertaa enemmän aikaa, kuin suunnitelmassa.

Ajankäytön arviointi oli vaikeaa, sillä osa asioista osoittautui huomattavasti työläämmäksi, mitä olin kuvitellut.

Suunnitelman toteutusjärjestys piti aika hyvin paikkansa. Lähdin liikkeelle GUI luokasta, sekä Tile ja tileGrid luokista. Esimerkiksi GUi luokka oli kuitenkin aika suuri, ja lähes koko projekti pyöri sen ympärillä.

13. **Toteutunut työjärjestys ja aikataulu**

    Aloitin projektin tekemällä GUI luokkaa, sekä Tile ja tilegrid luokkaa. Aloitin projektin noin 2 viikkoa enne ensimmäistä välipalautusta. Noin 10.3.

    Graphic item, sekä player luokat luotiin tässä välissä.

    Napeille tein omat luokat, joka haarautui Guista. tämä pushattiin 12.4. Näiden aikojen välissä keskityin lähinnä GUI luokkan hiomiseen.

    attack, effect, read_file ja test luokat luotiin 13.4.

    Lopuksi tehtiin Turn luokka ja AI luokat.

    Järjestyksessä noudatettiin aika pitkälti suunnitelmaa.

14. **Arvio lopputuloksesta**

    Ohjelma on pelinä mielestäni hyvä, mutta koodin on aika sotkuista. Työn suurimmat puuteet liittyvät testaukseen ja kaatumisen estoihin yms. Tämä pääasiassa johtuu siitä, että aika loppui hyvin rehellisesti kesken. Projetkin laajuus lopulta iski vasten kasvoja, ja resurssit eivät riittäneet. Ratkaisumenetelmät, luokkajaot olivat ihan ok. Suurin virhe, oli ehkä kunnianhimoisuudessa. Yksinkertaisempi peli, ja keskittyminen laadukkaaseen koodiin, olisi ollut jälkiviisaana parempi ratkaisu.

    Ohjelman positiivinen puoli on, että se on pelattavissa. Peli näyttää miellyttävältä, iskut toimivat ja tekoäly tekee oikeita isku valintoja. Pelin pystyy pelata alusta loppuun useissa tapauksissa.

    Ohjelman heikkoudet ovat ehdottomasti koodin siisteydessä, sekä testauksissa ja kaatumisen ennakoimisessa. Ohjelman ei pitäisi kovin usein kaatua, mutta se ei sisällä minkään laisia varatoimia tätä vastaan.

    Ohjelman luonne mielestäni soveltuu laajennukselle. Esim. eri hahmojen lisäys onnistuu kohtuullisen helposti.

15. **Viitteet**

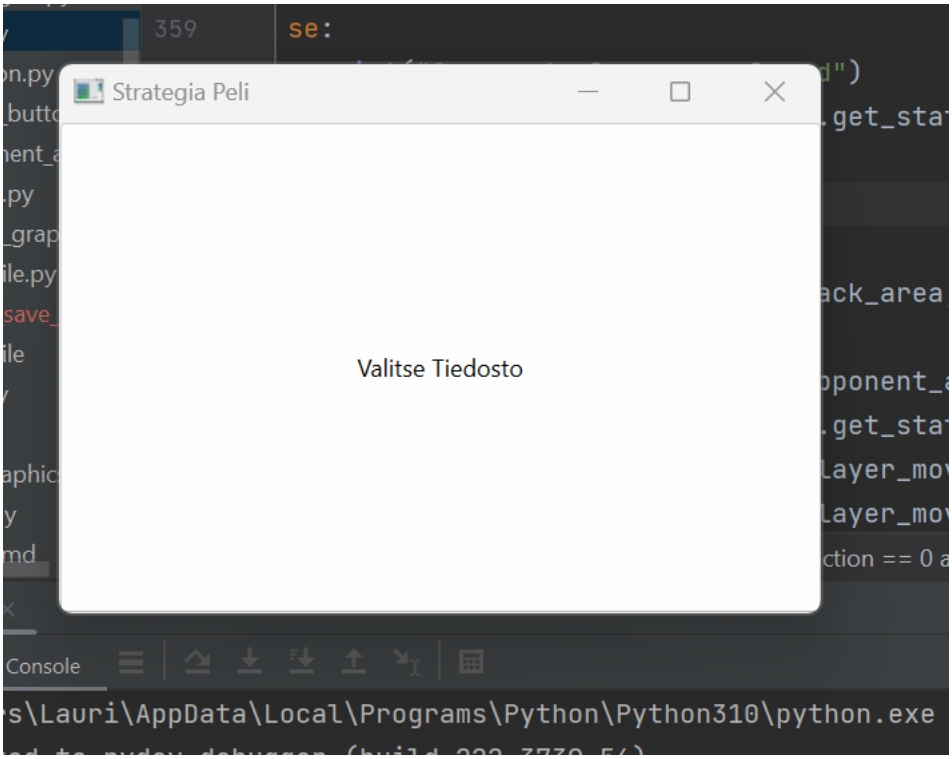
[QFileDialog - Qt for Python]

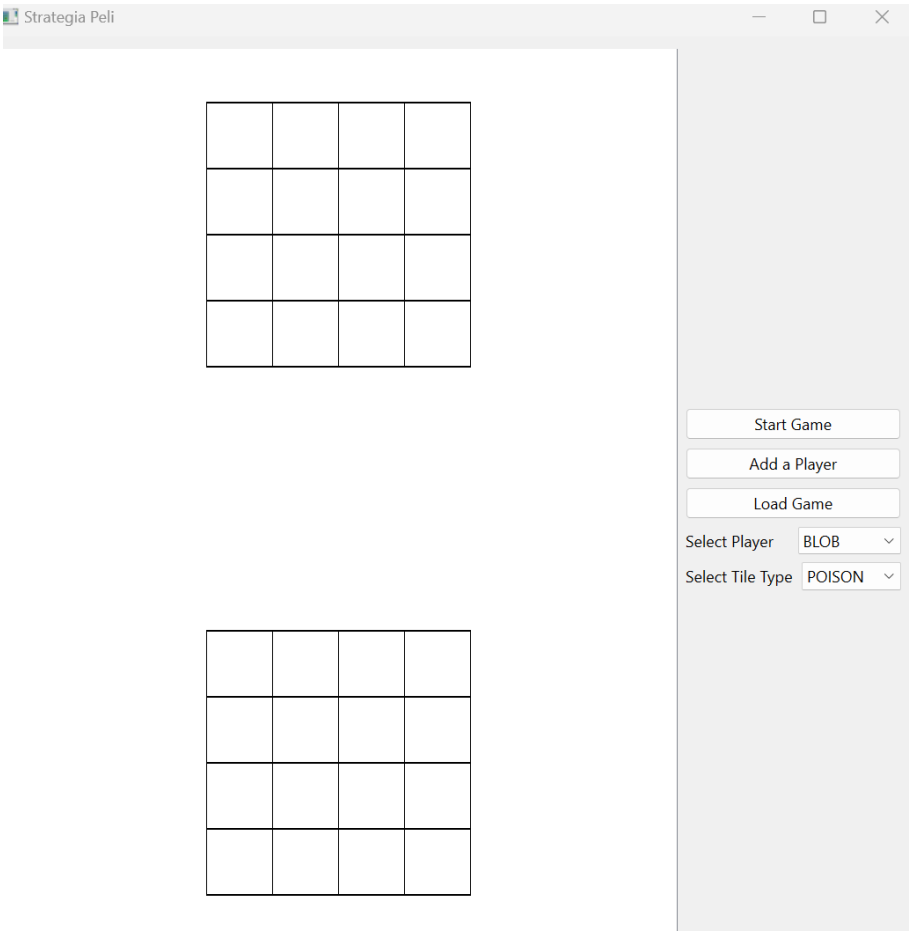
[Qt for Python]

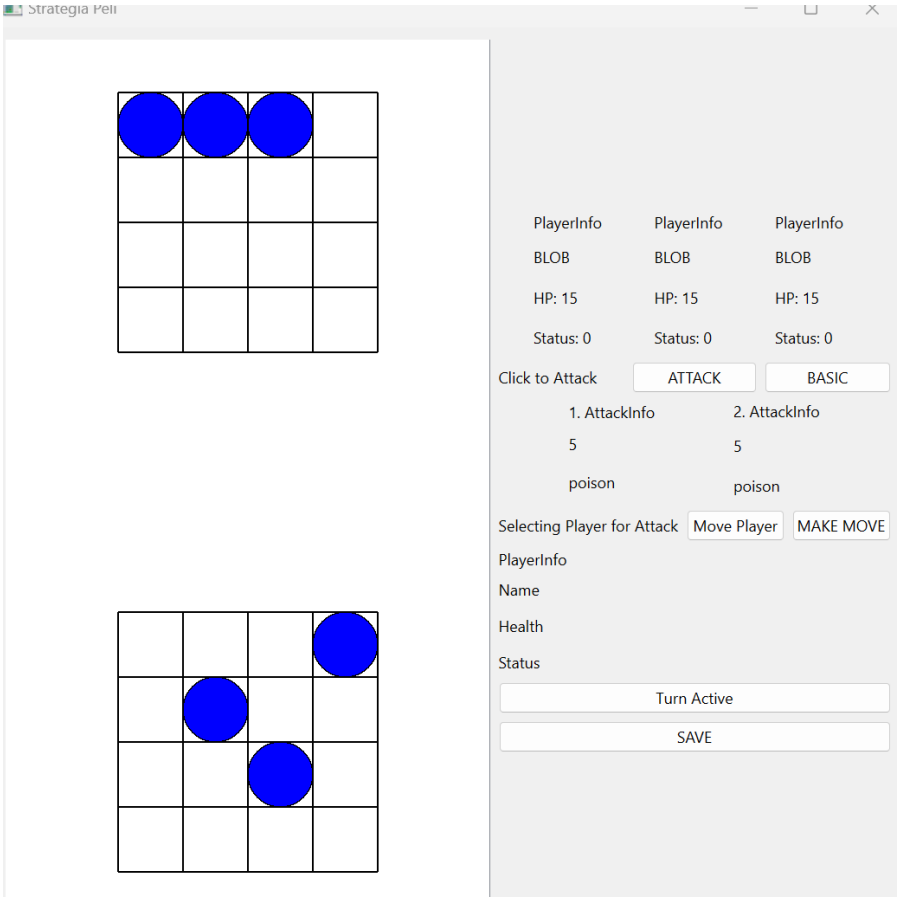
Kurssin CSA-1121 esimerkki Robot World

16. **Liitteet**

Kuvia pelistä.




Strategia Peli
Start Game
Add a Player
Load Game
Select Player
BLOB
Select Tile Type
POISON


PlayerInfo
BLOB
HP: 15
Status: 0
PlayerInfo
BLOB
HP: 15
Status: 0
PlayerInfo
BLOB
HP: 15
Status: 0
Click to Attack
ATTACK
BASIC
1. AttackInfo
5
poison
2. AttackInfo
5
poison
Selecting Player for Attack
Move Player
MAKE MOVE
PlayerInfo
Name
Health
Status
Turn Active
SAVE