**Y2 Strategiapeli – dokumentaatio 1.5.2020**

**Henkilötiedot**

**Nimi Ville Synkkänen Opiskelijanumero 710688 Koulutusohjelma Automaatio- ja informaatioteknologia Vuosikurssi 2019**

**Yleiskuvaus**

**Projekti on vuoropohjainen strategiapeli, jossa pelataan tietokonevastustajaa vastaan. Pelikenttä on ruudukkopohjainen. Pelissä komennetaan joukkoa erilaisia yksiköitä, joilla on erilaisia vuoron aikana suoritettavia toimintoja, kuten erilaisia hyökkäyksiä ja muita kykyjä, jotka voivat aiheuttaa omiin ja vastustajan yksiköihin erilaisia vaikutuksia. Pelissä on erilaisia kenttiä, joissa on eri tavoin yksiköihin vaikuttavia maastoja. Pelitilanne on mahdollista tallentaa tiedostoon, mikä mahdollistaa kesken jääneen pelin jatkamisen myöhemmin. Lisäksi pelissä on kenttäeditori, jonka avulla pystyy helposti luomaan uusia kenttiä tai muokkaamaan olemassa olevia kenttiä.**

**Ominaisuuksia**

* Graafinen käyttöliittymä
* Yksinkertaiset grafiikat
* Ruudukkopohjainen
* Omaan algoritmiin perustuva tekoäly
* A\*-polunhakualgoritmi
* Erilaisia yksikkötyyppejä, joilla useita kykyjä
  + energiasysteemi: kykyjen käyttäminen vaatii energiaa, jota yksikkö saa pelin edetessä
  + passiivisia kykyjä, jotka vaikuttavat yksiköiden ominaisuuksiin
* Erilaisia maastoja
* Näkyvyysjärjestelmä: jotkut maastot voivat estää näkyvyyden (ja näin hyökkäyksen)
* Eri kokoisia ja maastoltaan vaihtelevia pelikenttiä
* Konfiguroitavuus ulkoisten asetustiedostojen kautta
  + yksiköiden ominaisuudet
  + eri maastojen ominaisuudet
  + käyttöliittymän asetukset
* Peliloki, johon tulostetaan pelin tapahtumat
* Mahdollisuus tallentaa pelitilanne tiedostoon
* Kenttäeditori
  + uusien kenttien luonti
  + olemassa olevien kenttien muokkaus

**Käyttöohje**

**Ohjelma käynnistetään ajamalla main.py-tiedosto komentoriviltä tai IDE:stä käsin. Ohjelma avautuu päävalikkoon. Päävalikon kautta pääsee jatkamaan tallennettua peliä (jos sellainen on), valitsemaan pelattavaa kenttää tai kenttäeditoriin. (selosta tämä readme-tiedostossa)**

**Kenttäeditorissa on aluksi valittavana kaksi vaihtoehtoa: uuden kentän luominen ja olemassa olevan kentän muokkaaminen. Uusi kenttä luodaan syöttämällä kentän mitat ensimmäiseen tekstilaatikkoon välilyönnillä erotettuna ja painamalla UUSI KENTTÄ-nappia. Olemassa olevaa kenttää pääsee muokkaamaan syöttämällä kentän nimi ilman tiedostopäätettä toiseen tekstilaatikkoon ja painamalla MUOKKAA KENTTÄÄ-nappia. Tämän jälkeen ruudulle piirtyy kenttä. Kentälle voi lisätä elementtejä valitsemalla nappia painamalla haluamansa elementti ja klikkaamalla kentällä olevaa ruutua. Erityyppisten elementtien välillä voi vaihtaa painamalla VAIHDA YKSIKÖT/MAASTO-nappia. Kentän voi tyhjentää painamalla TYHJENNÄ KENTTÄ-nappia. Jos kentän haluaa tallentaa, se tapahtuu syöttämällä kentän nimi ilman tiedostopäätettä ensimmäiseen tekstilaatikkoon ja painamalla TALLENNA KENTTÄ-nappia. Olemassa olevan kentän tapauksessa voi joko tallentaa kentästä uuden version eri nimellä tai tallentaa vanhan kentän päälle syöttämällä vanhan kentän nimen tekstilaatikkoon. Editorista voi poistua tallentamatta kenttää painamalla POISTU EDITORISTA-nappia**

**Painamalla PELAA-nappia päävalikossa pääsee valitsemaan haluamansa kentän. Kenttiä voi selata käyttöliittymän napeilla. Kentän voi valita painamalla VALITSE-nappia, jolloin peli lataa valitun kentän. Valikosta pääsee takaisin päävalikkoon painamalla POISTU VALIKOSTA-nappia. JATKA-napin avulla pääsee suoraan tallennettuun peliin. Tallennettuja pelejä voi olla vain yksi kerrallaan.**

**Pelinäkymässä pelaaja voi valita yksikön klikkaamalla sitä, jolloin yksikkö muuttuu väriltään hieman kirkkaammaksi. Yksikön liikuttaminen tapahtuu klikkaamalla ruutua, joka on yksikön kantaman sisällä. Ruudut, joihin on mahdollista liikkua, on värjätty tummemmalla värillä kuin muut ruudut. Jos yksikkö pystyy hyökkäämään jonkin vihollisen kimppuun, pelaaja voi painaa HYÖKKÄÄ-nappia, jolloin mahdolliset hyökkäyksen kohteet muuttuvat väriltään kirkkaammiksi ja kantamalla olevat ruudut muuttuvat hieman tummemmiksi. Pitämällä hiirtä kohteen yläpuolella näkyviin tulee yksikön tiedot ja hyökkäyksen odotettu tulos. Klikkaamalla vihollista yksikkö hyökkää vihollisen kimppuun. Hyökkäyksen tai kyvyn käyttämisen voi perua painamalla PERU KOHTEEN VALINTA-nappia. Kun yksikkö ei pysty enää tekemään mitään tällä vuorolla, se muuttuu väriltään tummemmaksi (kun yksikön valinta poistetaan).**

**Yksikön kyvyistä saa lisätietoa valitsemalla yksikön ja painamalla YKSIKÖN TIEDOT-nappia, jolloin pelilokin paikalle ilmestyy kuvaus yksikön kyvyistä. Samat tiedot saa näkyviin pitämällä hiirtä kykynapin päällä. Pelilokin saa takaisin näkyviin painamalla PELILOKI-nappia. Lisäksi maastoista ja yksiköistä saa lisätietoa pitämällä hiirtä niiden yläpuolella. Pelaaja voi selata toimivia yksiköitään käyttöliittymän napeilla. Yksikön valinnan voi poistaa painamalla POISTA YKSIKÖN VALINTA-nappia. Pelaaja voi päättää vuoronsa painamalla PÄÄTÄ VUORO-nappia. Pelistä voi poistua joko painamalla TALLENNA JA POISTU-nappia tai tallentamatta painamalla POISTU PELISTÄ-nappia**

**Pelin edetessä yksikkö saa energiaa, jota se voi käyttää kykyihin. Jos yksikkö pystyy käyttämään kyvyn, kykynappi on aktiivinen. Nappia painamalla pääsee joko valitsemaan kohdetta kyvylle tai painallus aktivoi kyvyn suoraan riippuen kyvystä. Jos yksikkö saa tilavaikutuksen, sen elämäpalkki muuttaa väriään. Pitämällä hiirtä palkin yläpuolella tilavaikutuksesta saa lisätietoa.**

**Ulkoiset kirjastot**

**Käyttöliittymän toteuttamiseen on käytetty PyQt5:ä.**

**Ohjelman rakenne**

----luokkakaavio, selostus---

**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 käytätte 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, liittäkää mukaan jonkinlainen graafinen luokkakaavio (voitte käyttää esim. UML-luokkakaavionotaatiota, mutta se ei ole millään muotoa pakollista). Esitelkää omien luokkien keskeiset metodit. Huom. oleellista on vain se, mitä metodeilla tehdään, ei se, miten ne sisäisesti toimivat.

**Algoritmit**

-polunhaku (A\*)

-näkyvyys

-tekoäly (yleiset, mainitse lisäksi, että on yksiköille ominaisia)

-vahingon laskenta

-muuta?

**Algoritmit**

Sanallinen kuvaus käyttämistänne algoritmeista, eli siitä miten ohjelma suorittaa tarvittavat tehtävät. Esim. miten tarvittava matemaattinen laskenta tapahtuu? (kaavat mukaan) Miten algoritminne löytää lyhimmän tiereitin kahden kaupungin välille? Miten toteuttamanne pelin tekoäly toimii? Kaavioita tms. voi käyttää apuna tarpeen mukaan. Mitä muita ratkaisuvaihtoehtoja olisi ollut? Perustelkaa valintanne: Verratkaa toteutusta johonkin toiseen ratkaisuun, ja selittäkää miksi päädyitte juuri tähän.

Tässä kohdassa on siis tarkoitus selostaa ne periaatteet, joilla ongelmat on ratkaistu, ei sitä, miten algoritmit koodataan. Siis ei luokkien tai metodien kuvauksia tai muitakaan Pythoniin tai ohjelmakoodiin liittyviä seikkoja tänne. Pseudokoodiesitys keskeisimmistä ei-tunnetuista algoritmeista on kuitenkin hyvä olla sanallisen kuvauksen tukena. **HUOM!** Jokaisessa työssä on aina algoritmeja, toiset ehkä yksinkertaisempia kuin toiset, moni aivan itse alusta saakka keksittyjä. Kuvaa tässä niistä muutama kaikkein olennaisin.

**Tietorakenteet**

**Ohjelmassa on käytetty monipuolisesti erilaisia tietorakenteita. Tapauksissa, joissa on tarvittu muuttuvia rakenteita, on käytetty listoja tai joissain tapauksissa sanakirjoja. Listoja on käytetty muun muassa pelikentän ruutujen, yksiköiden, tilavaikutusten ja hyökkäyksen mahdollisten kohteiden käsittelyyn, koska näitä tulee pystyä muuttamaan pelin aikana. Ruutujen säilömisessä olisi voinut periaatteessa käyttää tupleja, mutta kenttäeditoria varten olisi joka tapauksessa tarvinnut muuttuvan rakenteen. Käyttämällä listoja pystyttiin käsittelemään ruutuja samalla tavalla kuin pelin aikana ja säästyttiin ylimääräisen koodin kirjoittamiselta. Sanakirjoja on käytetty muun muassa karttojen tietojen tallentamisessa ja asetustiedostojen tietojen käsittelyssä, jotta tiedostoista luettuja muuttujia pystytään käyttämään mahdollisimman helposti. Lisäksi polunhakualgoritmi käyttää sanakirjoja ruutuun liikkumiseen vaadittavan pistemäärän tallentamiseksi. Tuplejen käyttö on jäänyt ohjelmassa vähemmälle. Niitä on käytetty erilaisissa tilanteissa, joissa on varmaa, että tietoja ei tarvitse muuttaa myöhemmin. Hyvä esimerkki tästä on tiedostosta luettujen karttojen tiedot, jotka tallennetaan kukin omaan tupleensa, johon sisältyy kartan mitat, yksiköt ja ruudut.**

**Ohjelmassa on lisäksi käytetty useita omatekoisia tietorakenteita. Koordinaatit-luokka esittää nimensä mukaisesti koordinaatteja x, y-koordinaatistossa. Prioriteettijonon elementti sisältää varsinaisen elementin ja sen prioriteettia kuvaavan luvun. Polunhakujono on jono, joka sisältää edellä mainittuja jonon elementtejä. Jonoa käytetään polunhaussa lyhyimmän polun rakentamiseen. Jonoon on mahdollista lisätä elementti tai poistaa ja palauttaa elementti, jolla on suurin prioriteetti (pienin prioriteettiluku). Vaihtoehtoisesti polunhaussa olisi voinut käyttää Pythonin omaa heapq-moduulia, mutta se ei sopinut yhteen polunhaun toteutustapani kanssa, joten jouduin kehittämään korvaavan tietorakenteen. Lisäksi tilavaikutukset ja yksikön ominaisuudet on sisällytetty kummatkin omaan tietorakenteeseensa. Tilavaikutukset-rakenne sisältää tiedon tilavaikutuksen omaavasta yksiköstä, vaikutuksen kestosta ja muista vaikutukseen liittyvistä muuttujista. Vastaavasti yksikön\_ominaisuudet-rakenne sisältää tiedot yksikön tyypistä ja yksikön ominaisuudet määrittelevistä muuttujista. Omatekoisten rakenteiden sijaan olisi ollut mahdollista käyttää tupleja tai listoja, mutta omien rakenteiden käyttö on varsinkin koodin luettavuuden kannalta paljon selkeämpää.**

**Tiedostot**

**Ohjelman käsittelemät tiedostot voidaan jakaa kolmeen päätyyppiin: tallennustiedostoihin, asetustiedostoihin ja karttatiedostoihin. Kaikki tiedostot ovat tekstitiedostoja.**

**Ohjelma tallentaa pelitilanteen pelitilanne-kansiossa sijaitsevaan pelitilanne.txt-tiedostoon. Tiedoston alussa on kerrottu kentän nimi tiedostopäätteen kanssa. Tämän jälkeen tiedostossa on listattu yksiköt jokainen omalla rivillään. Yksikön tiedot ovat pilkuilla erotettuna järjestyksessä: x-koordinaatti, y-koordinaatti, omistaja (PLR tai COM), tyyppi, elämä, energia, liikkuminen käytetty, hyökkäys käytetty, loppuvaikutus ja tilavaikutukset. Tilavaikutuksessa on listattu sanan tilavaikutus ja kaksoispisteen jälkeen vaikutukseen liittyvät parametrit järjestyksessä: kesto, hyökkäysbonus, puolustusbonus, liikkumisbonus, verenvuoto, taintuminen ja loppuvaikutus, jonka jälkeen tulee puolipiste. Yksiköiden jälkeen omalla rivillään on listattu kiilojen paikat kentällä. X- ja y-koordinaatit on erotettu pilkuilla ja kiilat on erotettu kaksoispisteillä. Tiedoston viimeisellä rivillä on sana LOPPU. Käyttäjän on mahdollista halutessaan muokata tiedostoa. Dokumentaatio-kansiossa on esimerkkitiedosto, joka havainnollistaa, miltä tiedostoformaatti näyttää.**

**Ohjelman asetustiedostot on tallennettu maastot-, yksikot- ja muut-kansioihin. Maastot-kansio sisältää maastoihin liittyvät asetustiedostot. Jokaisessa tiedostossa on listattu maaston tyyppi, liikkuminen, liikkumisen hinta, puolustus- ja hyökkäyskertoimet, läpinäkyvyys ja ruudun väri RGB-muodossa kukin omalla rivillään. Tiedoston viimeisellä rivillä on sana LOPPU. Yksiköt-kansio sisältää vastaavasti yksiköihin liittyvät asetustiedostot. Tiedostojen formaatti on hyvin samankaltainen kuin maastoilla. Jokaisesta yksiköstä on listattu tyyppi, liikkumispisteet, maksimielämä ja -energia, hyökkäys, puolustus, kantama ja pistehinta (ei tee mitään pelin nykyisessä versiossa) sekä yksikön kykyihin liittyvät parametrit, jotka ovat jokaiselle yksikölle yksilöllisiä. Muut-kansiossa on käyttöliittymän asetustiedosto, jonka formaatti on myös muiden asetustiedostojen kaltainen. Tiedostossa on listattuna näytön resoluutio (erottimena \*-merkki), joka määrittää, mihin kohtaan näyttöä ohjelman ikkuna siirretään alussa, ikkunan koko ja tekoälyn toimintojen välinen viive.**

**Testaus**

**Ohjelmaa on testattu sekä kokeilemalla sen ominaisuuksia käytännössä että yksikkötestauksen avulla. Ohjelman ominaisuuksien toimintaa on testattu käytännössä kehityksen aikana sitä mukaa, kun ne ovat tulleet valmiiksi. Lisäksi ohjelman valmistumisen jälkeen käytiin vielä läpi kaikki ohjelman keskeiset osa-alueet ja varmistettiin niiden oikea toiminta. Käyttöliittymää testattiin kokeilemalla erilaisia syöteyhdistelmiä nappeihin ja pelikenttään.**

**Yksikkötestauksen avulla on testattu ohjelmassa tapahtuvaa laskentaa, tekoälyn yleisiä metodeja, näkyvyysjärjestelmää ja polunhakua. Laskentaa testattiin kutsumalla laskentaan liittyviä metodeja ja tarkastamalla, että ne palauttavat oikeanlaisia tuloksia. Tekoälyn metodeja testattiin siten, että tekoälylle annettiin erilaisia toimintavaihtoehtoja ja katsottiin, valitseeko se vaihtoehdoista parhaan. Jokaiselle yksikkötyypille yksilölliset tekoälyn osat jäivät ajan puutteen vuoksi testaamatta. Näkyvyyttä testattiin yksinkertaisesti laskemalla näkyvyys monessa eri tilanteessa ja tarkistamalla, että palautetut arvot olivat oikeita. Polunhakua testattiin laskemalla, kuinka monta pistettä johonkin ruutuun liikkumisessa kuluu ja tarkistamalla, ovatko palautetut arvot oikeita. Lisäksi testattiin tapauksia, joissa kohderuutu oli blokattuna.**

**Testaus vastasi muuten suunnitelmassa esitettyä, mutta tiedostojen lukemisen testaus tapahtui yksikkötestien sijaan testaamalla, miten ohjelma reagoi tiedostojen muokkaamiseen. Tein näin siksi, että sain käsin testaamalla hyviä tuloksia, eikä yksikkötestaus olisi tuonut testaukseen merkittävää lisäarvoa.**

**Puutteet ja viat**

**-bugi(t)**

**-käyttöliittymän puutteet, peli voi olla hankala seurata**

**-tekoälyn optimisaatio ja toiminta (joissain tilanteissa)**

**Ohjelman tunnetut puutteet ja viat** Kuvaa tässä osiossa kaikki tuntemasi puutteet ja viat ohjelmassasi. Kerro miten korjaisit nämä ongelmat jos jatkaisit projektia. Mitä vähemmän assistentti löytää puutteita kohdista, joiden väität toimivan sen parempi. Ole siis rehellinen. Lisäksi hyvin jäsennetyt kehitystarpeet kertovat perehtymistä ongelmaan ja sen ratkaisuun, sekä kriittisestä oman työn arvioinnista.

**Parhaat ja heikoimmat kohdat**

**3 parasta ja 3 heikointa kohtaa** Assistentti käyttää runsaasti aikaa tutustuessaan ohjelmaasi, mutta ei välttämättä näe ja tunne toteutustasi samalla tavoin kuin sinä. Jos ohjelmassa on joitakin kohtia joita itse pidät erityisen hyvinä, mainitse tässä niistä 1-3 kappaletta lyhyen perustelun kera. Jos ohjelmassa on kohtia, jotka itsekin tiedät heikoiksi, voi mainita myös nämä. Tällöin mahdollisuus, että nämä heikot kohdat dominoivat arvostelua, vähenee huomattavasti. Tässä voi myös esittää sanallisesti kuinka olisi nämä asiat halutessaan korjannut.

**Poikkeamat suunnitelmasta**

-yksiköiden asettelu pois, toteutusjärjestyksen muutokset

meni enemmän aikaa kun olin arvioinut (100-120 h)

**Poikkeamat suunnitelmasta**

Teitkö jotain toisin kuin olit suunnitellut? Miksi? Osuiko suunnitelmaan laatimasi ajankäyttöarvio oikeaan? Miksei? Entä toteutusjärjestys?

**Työjärjestys ja aikataulu**

-ks. toteutusjärjestys commiteista, poikkeamat

**Toteutunut työjärjestys ja aikataulu** Kerro tässä yleisellä tasolla missä järjestyksessä projekti lopulta toteutettiin (mielellään myös päivämäärät). Missä poikettiin suunnitelmasta?

**Arvio lopputuloksesta**

**-ok**

**Arvio lopputuloksesta**

“Yhteenveto” ja itsearviointi joka voi toistaa yllämainittujakin asioita.

Arvioikaa ohjelman laatua, kertokaa sen hyvistä ja huonoista puolista. Onko työssä oleellisia puutteita ja mistä ne johtuvat (mahdollinen hyvä perustelu dokumentissa voi korvata pienet puutteet)? Miten ohjelmaa olisi voinut tai voisi tulevaisuudessa parantaa? Olisiko ratkaisumenetelmien, tietorakenteiden tai luokkajaon valinnan voinut tehdä paremmin? Soveltuuko ohjelman rakenne muutosten tai laajennusten tekemiseen? Miksi tai miksi ei?

**Viitteet**

**-ks. tekninen suunnitelma**

**Liitteet**

**-ota screenshotteja**

**-pelimekaniikat (txt tiedosto)**

**-muuta?**

**Liitteet**

Liitteeksi tulee mahdollisten muiden liitteiden lisäksi ainakin tekstipohjaisissa ohjelmissa laittaa **muutama havainnollinen ajoesimerkki**, jotka on kätevä tehdä unixympäristöissä script-ohjelmalla. Script-ohjelma käynnistyy kirjoittamalla script ja päättyy painamalla CRTL-D tai unix shellissa exit. Scriptin ollessa käynnissä se tallentaa tiedostoon kaiken ruudulle ilmestyvän sekä käyttäjän kirjoittaman syöteen. Graafisissa töissä ajoesimerkkejä ei vaadita, mutta muutama todellinen kuva ohjelman käytöstä joko erillisenä liitteenä tai käyttöohjeen yhteydessä ei tekisi pahaa. (Kuvia voi Linux- tai Windows ympäristöissä napsia painamalla Alt+PrintScreen-nappuloita, (Riippuen käyttöjärjestelmästä voit antaa joko tallenettavan kuvan nimen heti tai kuva on leikepöydällä, mistä sen voi tallentaa jonkin piirto-ohjelman kautta.)