Kasino peli Y2

1. **Henkilötiedot**

   Eero Saarro, 710 222, Bioinformaatioteknologia, 05/05/2021

2. **Yleiskuvaus**

   Projektin aiheena oli tehdä Kasino niminen korttipeli, jota pystyy pelaamaan graafisen käyttöliittymän avulla. Projektissa tein niin sanotun pakkakasino-pelin, jossa pyritään keräämään kierroksen aikana kortteja sekä estää toisia pelaajia saamasta helppoja pisteitä. Jokaisen kierroksen lopussa lasketaan pisteet kierroksen aikana saatujen korttien perusteella ja usein pelataan useita kierroksia. Ensimmäisenä 16 pistettä saanut pelaaja voittaa pelin.

   Päädyin tekemään pelistä keskivaikean version, jossa on graafinen käyttöliittymä sekä mahdollisuus tallentaa ja ladata pelitilanne täsmällisesti. En ehtinyt tekemään vaikeassa versiossa vaadittua tietokonepelaajaa, mutta suunnitelmaan nähden lisäsin peliin mahdollisuuden laittaa musiikin päälle ja pois sekä katsoa sen hetkisen pistetilanteen uudesta ikkunasta.

3. **Käyttöohje**

   Ohjelma ja peli käynnistetään ajamalla #main.py#-tiedostoa. #Main.py" tiedoston ajaminen avaa aloitusnäytön, josta voi valita haluaako aloittaa uuden pelin vai ladata edellisellä pelikerralla tallennetun tiedoston. Uuden pelin aloittaminen pyytää käyttäjää kertomaan pelaajien lukumäärän sekä antamaan pelaajille nimet. Näyttöön on lisätty menubar-valikko, josta voi avata pistetilanneikkunan, tallentaa pelin, laittaa musiikin päälle sekä pois sekä lopettaa pelin. Pelitilannetta sekä ohjeita pystyy seuraamaan statusbarin avulla ikkunan alareunassa.

   Pelin ollessa käynnissä avautuu näytölle pelipöytä, jossa on pöydällä olevat kortit näkyvisssä, sekä pelaajan käsi, joka on aina vuoron alettua piilossa. Pelaajan, jonka vuoro on sillä hetkellä, täytyy ensiksi avata omat korttinsa painamalla näytä kortit näppäintä. Tämän jälkeen käyttäjä voi tehdä siirtonsa valitsemalla kädestä sekä halutessaan pöydästä haluamansa kortit ja siten painamalla "ota valitut kortit" tai "laita kortti pöytään painikkeita.

   Kierroksen loputtua käyttäjä voi aloittaa uuden kierroksen painamalla silloin ruudulle ilmestyvää "aloita uusi kierros" näppäintä, jos kenelläkään pelaajalla ei ole vielä yli 16 pistettä.

4. **Ulkoiset kirjastot**

   Projektissa olen käyttänyt pyqt5 kirjastoa graafisen käyttöliittymän tekemiseen sekä random kirjastoa sekoittamaan pakan. Päädyin käyttämään myös viime

hetkillä Itertools kirjastosta funktiota combinations, sillä havaitsin virheen laske oikein algortimissa.

5. **Ohjelman rakenne**

   Ohjelma noudattaa pitkälti suunnitelman UML-luokkakaavionotaatiota. Ohjelma on jaettu niin, että osa luokista vastaavat graafisesta käyttöliittymästä ja toiset eivät sisällä Pyqt5 kirjaston ohjelmia. Ohjelman keskeisin luokka on ”Qui_aloitus”, joka perii QMainwindow olion Pyqt5 kirjastosta. Tämä luokka vastaa pelin graafisen käyttöliittymän luomisesta, ja siten se luo kaikki metodit, joilla peliä pelataan. ”Qui_aloitus” luokan kautta luodaan myös muut luokat, joten se on keskeinen osa pelin toimintaa. Muita graafisen ulkoasuun ja pyqt5 kirjastoa hyödyntäviä luokkia ovat ”Qui_pisteet”, joka avataan ”Qui_aloitus” luokasta ja joka esittää pistetilanteen pelissä sekä ”Qui_card” luokka, joka luo pelikorteille graafisen ulkoasun ja lataa niille kuvan. ”Qui_card” hyödyntää QLabel widgettiä PyQt5 kirjastosta ja sen yksi tärkeimmistä metodeista on ”Mousepressevent”, mikä mahdollistaa korttien klikkaamisen. Korttien sijainnista pöydällä sekä liikuttamisesta vastaa myös ”Qui_aloitus” luokka, joka myös luo ”Qui_card” widgetit.

   Pelin toinen keskeinen luokka on ”Pelikenttä”, joka vastaa pelin ylläpidosta. ”Pelikenttä” luokassa löytyy metodit pelaajien lisäämiseksi peliin eli Pelikenttä luokan listaan. Pelaajia esittää Player luokka, joka pitää yllä tietoa pelaajan pisteistä sekä kädessä olevista korteista. ”Pelikenttä” luo myös uuden korttipakan eli ”Korttipakka” olion, jonne lisätään kierroksen alussa 52 pelikorttia eli ”Kortti” oliota. Pelikenttä pitää huolta pöydällä olevista korteista ja jokainen pelaaja kädessä olevista korteista.

6. **Algoritmit**

   Ohjelman tärkein algoritmi on ”Pelikenttä” luokassa oleva laske oikein metodi. Tämä algoritmi laskee, onko pelaajan tekemä siirto oikein eli vastaako kädestä valitun kortin arvo, pöydältä otettujen korttien arvojen summaa. Tämä onnistuu Iterations kirjaston combinations funktion avulla. Algoritmi katsoo ensiksi helpot tilanteet käymällä lävitse pöydältä valitut kortit. Algoritmi laskee näiden korttien summan ja jos kortit ovat valmiiksi oikeassa järjestyksessä nollaa summan aina kun summa on yhtä suuri kuin kädestä valitun kortin arvo. Tässä vaiheessa katsotaan myös, onko yhdenkään pöydästä valitun kortin arvo suurempi kuin kädestä valitun kortin arvo ja palauttaa False jos näin on. Jos kortti on pienempi kuin käden kortin arvo, lisätään sen kortin arvo listaan lista. Jos tämän jälkeen summa on 0 niin funktio palauttaa True, sillä korttien summat täsmäävät. Jos jakojäännös on nolla, kun jaetaan pöydästä valittujen korttien summa kädestä valitun kortin arvolla, lähdetään tutkimaan korttien summia combinations funktion avulla. Jos jakojäännös ei ole nolla funktio ilmoittaa siirron olevan virheellinen palauttamalla False. Combinations funktion avulla jaetaan lista nimisessä listassa olevat luvut kaikkiin mahdollisiin yhdistelmiin, joiden pituus on

väliltä 2 ja listan pituus. Tämän jälkeen lisätään uuteen listaan ne yhdistelmät alkioiksi, joiden summa on yhteensä yhtä suuri kuin kädestä valitun kortin. Tämän jälkeen jaetaan uusi lista uudestaan combinations funktion avulla kaikkiin mahdollisiin yhdistelmiin näiden edellä valittujen yhdistelmien välillä. Sitten verrataan näiden uusien yhdistelmien pituutta alkuperäiseen listaan ja valitaan ne yhdistelmät viimeiseen listaan, joiden pituus on yhtä suuri kuin alkuperäisessä listassa nimeltä lista. Viimeiseksi verrataan valittujen oikean pituisten yhdistelmien lukuja alkuperäiseen listaan ja jos jokainen listan alkio esiintyy yhdistelmässä, palautetaan True, jolla ilmoitetaan siirron onnistuminen. Jos ei sopivia yhdistelmiä löydy ei siirto ole oikein ja palautetaan False.

Toinen algoritmi ohjelmassa laskee pisteet jokaiselle pelaajalle kierroksen loputtua lisäämällä pelaajalle pisteet ässistä, mökeistä, erikoiskorteista sekä katsomalla kenellä pelaajista on eniten patoja sekä kortteja.

7. **Tietorakenteet**

   Ohjelmassa käytetään paljon listoja, joilla kuvataan muun muassa korttipakkaa sekä pöydässä ja kädessä olevia kortteja. Listat soveltuvat hyvin ohjelmaan, sillä niihin voi helposti lisätät sekä poistaa alkioita. Tämän lisäksi ohjelman listat ovat lyhyitä, joten listat ovat myös algoritmianalyysisesti tehokkaita. Pakka, joka koostuu 52 kortista, on pelin pisin lista, mutta siitä otetaan kortteja aina toisesta päästä, joten lista toimii erittäin tehokkaasti.

   Ohjelmassa on hyödynnetty myös sanakirjaa korttien maiden luomisessa.

8. **Tiedostot**

   Ohjelma luo peliä tallentaessa tekstitiedoston, johon tallennetaan täsmällisesti kaikki tiedot pelistä ja tilanteesta. Tämän lisäski ohjelma käsittelee jpg muotoisia kuvia sekä mp3 muotoisia musiikki tiedostoja, jotka on lisätty kuvat ja musiikki kansioon.

9. **Testaus**

   Ohjelmaa varten on luotu unittestejä, joiden avulla pystyi testaamaan luokkia sekä metodeita, jotka eivät käytä Pyqt5 kirjastoa. ”Testit.py” tiedostoon on luotu testit luokka, joka testaa unitteisteillä Pelikenttä, Kortti, Player sekä Korttipakka luokan tärkeimpiä metodeita sekä algoritmeja. Näillä unittesteillä testailtiin ohjelmaa jo sen rakennusvaiheessa. Erityisesti ohjelman teon alussa pystyttiin unittestien avulla varmistumaan siitä, että ohjelma toimi oikein. Näin testattiin muun muassa pakan luontia sekä korttien jakamista pelaajille ja pöytään ennen kuin graafinen käyttöliittymä valmistui. Unittestien avulla pystyi myös varmistumaan siitä, että laske oikein algoritmi toimii oikein.

   Graafista käyttöliittymää testattiin ajamalla main.py tiedostoa, sillä Unittestien teko Pyqt5 kirjastoja hyödyntäville luokille, on haastavaa. Ajamalla tiedostoa,

pystyttiin myös graafisen käyttöliittymän valmistutta varmistumaan siitä, että toiset luokat toimivat oikein.

Ohjelman testaaminen meni juuri niin kuin jo suunnitelmassa suunnittelin ja ohjelma läpäisee kaikki testit kun testit.py tiedostoa ajetaan. Uuden kierroksen sekä pelin loppumisen testaamisen helpottamiseksi kannattaa ladata pelit voitto.txt sekä ok.txt

10. **Ohjelman tunnetut puutteet ja viat**

Ohjelmassa ei pitäisi olla yhtäkään suurta bugia, mikä estäisi pelaamisen. Yksi puute ohjelmassa on se, että pelaajan nimen ensimmäinen merkki ei saa olla '#', sillä tämä kaataisi ohjelman tallennustiedoston lataamisen yhteydessä. Tämän takia ohjelmassa ei ole mahdollista aloittaa nimeä '#' merkillä, vaan näin tehdessä, ohjelma ei hyväksy pelaajan nimeä.

Tämän lisäksi, kun pelistä löytyy voittaja niin peli päättyy uuteen ikkunaan ja ohjelma täytyy sulkea ennen kuin uuden pelin voi aloittaa.

Viimeisenä havaitsin testien avulla laske oikein algoritmin laskevan siirrot tietyissä tilanteissa väärin. Pyrin korjaamaan tämän tekemällä algoritmin monimutkaisemmin, mutta tämän testaaminen jäi hieman kiireelliseksi.

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

Ohjelman parhaimmat puolet liittyvät sen toimivuuteen pelatessa peliä yhdellä koneella kaverin kanssa sekä ohjelman graafiseen ulkonäköön. Pelaajan vuoron alkaessa ohjelma piilottaa aina pelaajan kortit, joten toinen pelaaja ei näe niitä vierestä ja ehtii siirtyä sivuun. Painamalla ”näytä kortit” ohjelma kääntää auki sen pelaajan kortit, jonka vuoro on. Kun hän on tehnyt siirtonsa ja painanut ”Seuraava vuoro”, kääntyy uuden pelaajan kortit heti piiloon.

Toinen hyvä puoli ohjelmassa on sen ulkoasu sekä lisätoiminnot. Ohjelmaan on lisätty valikkoon musiikki painike, joka alkaa soittamaan loopilla teeman mukaista musiikkia. Tämän lisäksi valikosta (Menubar) pystyy avaamaan pistetilanteen painamalla ”Pistetaulukko” nappia. Pistetaulukon avaamaan Qui_pisteet luokkaan on lisätty jokaisen pelaajan saadut kortit, ässät, mökit, padat, pata2, ruutu10 sekä pisteet. Tämän lisäksi luokan graafinen ulkoasu sekä tekstit ottavat huomioon sen, että onko kierros loppunut vai onko kierros vielä kesken. Pisteet lasketaan aina kierroksen loputtua ja 16 pistettä ensimmäisenä saanut voittaa pelin. Ohjelma ottaa huomioon myös tasapeli mahdollisuuden, kun kahdella pelaajalla on saman verran pisteitä sekä yli 15 pistettä ja tulostaa graafiseen käyttöliittymään molempien pelaajien tiedot.

Tämän lisäksi ohjelman ulkonäköön on panostettu lisäämällä useita kuvia eri näkymiin ja luokkiin, säätämällä tekstien fontteja, kursoreita sekä sijaintia ja lisäämällä kuvien päälle vielä uusia widgettejä sekä kuvia auttamaan hahmottamaan pelin toimintaa.

Olen tyytyväinen siihen, miltä pelin ulkoasu näyttää. Ohjelma myös toimii pääasiassa erinomaisesti, enkä ole onnistunut löytämään uusia selkeitä bugeja.

Laske oikein algoritmi tuotti loppuhetkillä suurta päänvaivaa ja vaikka en onnistunut löytämään virheitä uudesta versiosta, on mahdollista, että pelistä löytyy vielä joitain pieniä bugeja johtuen projektin vaikeudesta sekä monimutkaisuudesta. Tämän lisäksi muita parannuskohteita voisi olla esimerkiksi mahdollisuus palata takaisin alkuvalikkoon pelin loppuessa ja siten käynnistää uusi peli ilman ohjelman sulkemista sekä tietokonepelaajan lisääminen peliin, mikä jäi kesken.

12. **Poikkeamat suunnitelmasta**

    Suunnitelmaan nähden suurimmat muutokset liittyivät Qui_board luokan poistamiseen sekä laske oikein algoritmin tekemiseen. Jouduin lisäämään myös useita metodeita sekä attribuutteja luokkiin, haasteiden ja pikku asioiden korjaamiseksi.

    Muuten kaikki meni hyvin suunnitelman mukaan, vaikka aikataulun kanssa tulikin haasteita, sillä graafisen käyttöliittymän tekoon meni huomattavasti kauemmin kuin olin ajatellut. Myös projektin ja erityisesti Kasino-peli aiheen vaikeus pääsi vähän yllättämään.

    Ohjelman toteutus ja kasaus eteni juuri siinä järjestyksessä kuin olin ajatellut.

13. **Toteutunut työjärjestys ja aikataulu**

    Aloitin ohjelman tekemällä projektiin luokat Pelikenttä, Korttipakka, Player sekä Kortti jo maaliskuun alussa. Aloitin tekemään ohjelmaa näistä luokista, sillä nämä luokat eivät vaatineet graafista käyttöliittymää. Tämän jälkeen tein näille tehdyille metodeille sekä funktioille unittestit, joiden avulla varmistuin, että ohjelma toimii oikein. Pian tämän jälkeen aloitin tekemään graafista käyttöliittymää, mikä osoittautui yllättävän haastavaksi. Tämän takia aikaa kului paljon graafisen käyttöliittymän metodien harjoitteluun sekä käytön opettelemiseen. Graafisen käyttöliittymän tekemisen aloitin maaliskuun lopussa ja erityisesti 16.4 lähtien sain kortit näkyviin pöydälle. Viimeiseksi tein Pistetilanne luokan, lataus ja tallennus metodit sekä viimeistelin ohjelman ulkoasun sekä toiminnan. Laske oikein algoritmin haasteet yllättivät lopuksi jo, kun luulin projektin olleen valmis.

14. **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?

15. **Viitteet**

    Mitä kirjoja, nettisivuja tai muuta materiaalia olette käyttäneet? Kaikki lähteet tulee ilmoittaa, vaikka niihin kuuluisivat pelkkä kurssilla käyttämänne oppikirja ja perusluokkakirjastojen API-kuvaus.

16. **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.)