Kasino

Oskari Mäkinen
30.4.2018

Yleiskuvaus

Projektin aiheena oli toteuttaa ohjelma, jolla voi pelata Kasino-korttipeliä. Pelattava versio on 'pakkakasino', jossa pelaaja ottaa aina vuoronsa päätteeksi uuden kortin pakasta. 52:n kortin pakalla maksimimäärä pelaajia on 12. Pakkakasinon lisäksi on toteutettu vaihtoehto pelata perinteistä Kasinoa (jaetaan aina 4 korttia kerrallaan) 2-4 pelaajalla.

Pelissä pelataan useita kierroksia, ja kunkin kierroksen lopussa annetaan pisteitä pöydästä nostettujen korttien mukaan. Ensimmäisenä 16 pistettä saavuttanut pelaaja voittaa. Käyttäjä voi myös halutessaan muokata pisterajaa.

Työ on toteutettu vaikeimmalla vaatimustasolla, sisältäen graafisen käyttöliittymän, tekoälyn eri vaikeusasteilla, sekä tallennusmahdollisuuden milloin tahansa pelin aikana.

Käyttöohje

Ohjelman käyttämiseksi koneella tulee olla asennettuna Python 3 sekä PyQt5. Ohjelma käynnistyy ajamalla ’main.py’ -tiedosto. Esim. Windowsin komentoriviltä: cd <ohjelman hakemistopolku> → python main.py

Pelissä on 3 pääasiallista näkymää, ja kussakin on erilaisia toimintoja:

Aloitusnäkymä

- Start Game – aloittaa uuden pelin
- Load Game – jatka tallennettua peliä
- Quit – sulje ohjelma

Uuden pelin luomisnäkymä

- Auto Fill – täyttää automaattisesti tyhjät kohdat ja asettaa ne tietokonevastustajiksi
- Clear All – tyhjentää kaikki pelaajakentät
- Start – aktivoituu kun kaikki kohdat täytetty
- Auto – luo pelaajalle automaattisesti nimen
- Computer – rastita tehdäksesi pelaajasta AI-vastustajan, vaikeusasteen voi valita viereisestä valikosta
- Valikoista voi valita myös pelaajien määrän, pisterajan, sekä pelimoodin (Deal-1 on Pakkakasino, Deal-4 on perinteinen Kasino)

Pelinäkymä

- Painikkeet vaihtuvat tilanteen mukaan.
- Start Round – aloita kierros
- Make Move – yrittää suorittaa siirron pelaajan valitsemilla korteilla. Siirron onnistuessa vuoro siirtyy seuraavalle
- Pelissä on kaksi korttinäkymää, ylhäällä pöytä ja alhaalla käsikortit. Kortteja voi valita niitä klikkaamalla.
- Show Cards - Jos pelissä on useampi ihmispelaaja, saa käsikortit näkyviin tästä painikkeesta
- Ruudun oikealla puolella on loki, johon kirjataan pelin tapahtumia. Lokin alapuolella on pelaajien tiedot. Keltainen nuoli osoittaa kenen vuoro nyt on.

Ruudun ylälaidan valikkopalkista voi lisäksi valita animointinopeuden, sekä asettaa musiikin päälle tai pois (myös ctrl + M).

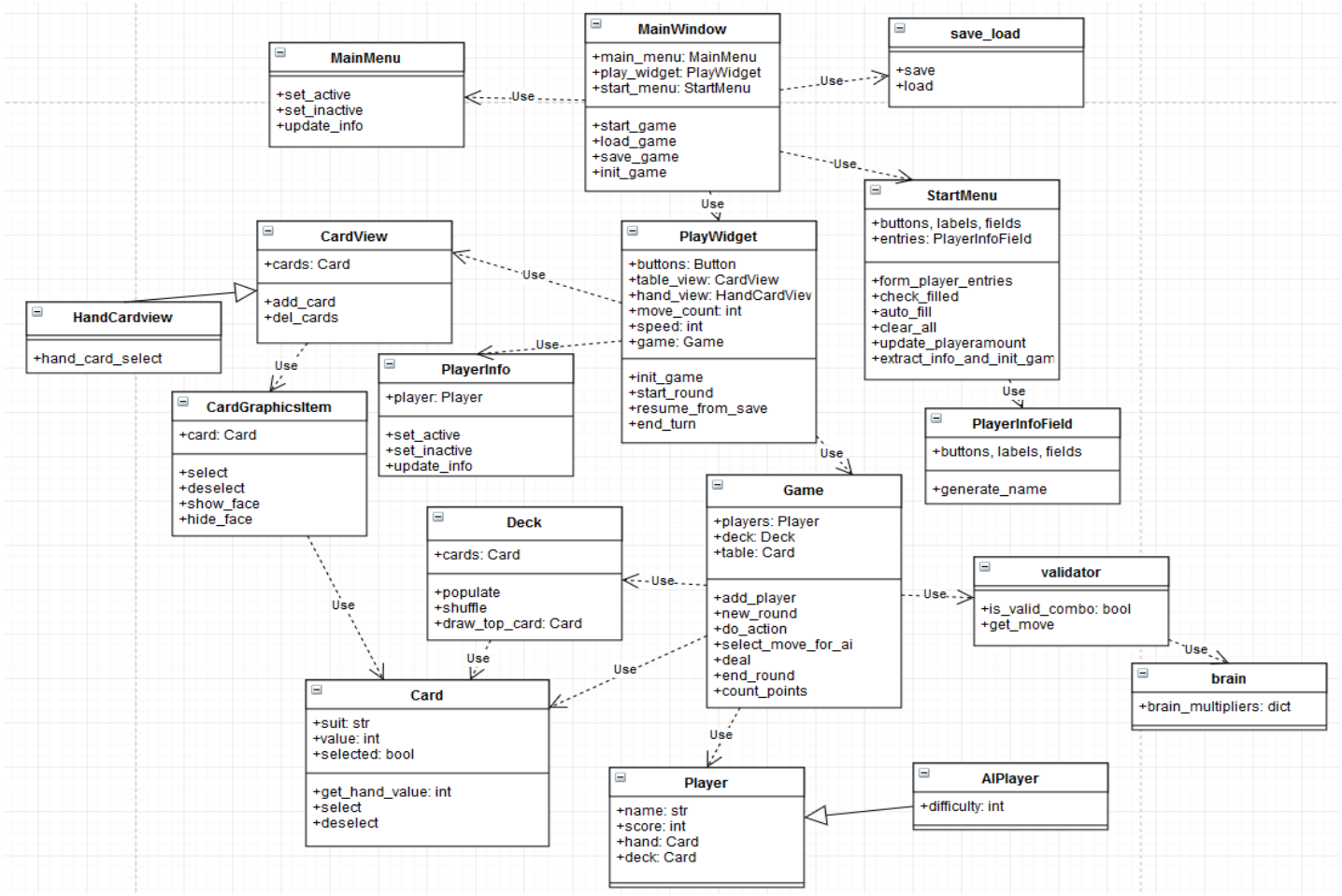
Ulkoiset kirjastot

Ainoa projektissa käytetty ulkoinen kirjasto on PyQt5. Tätä tarvitaan graafisen käyttöliittymän luomiseen.

Lisäksi käytössä on Pythonin sisäisistä kirjastoista ’random’ korttipakan sekoittamiseen sekä ’itertools’ tekoälyn ja tarkistusalgoritmin tehokkaaseen toteuttamiseen.

Ohjelman rakenne

Tässä toteutuneen ohjelman luokkakaavio. Pienellä kirjoitetut yksilöt ovat moduleita, CamelCasella kirjoitetut luokkia. Kaavioon on merkitty vain keskeisimmät attribuutit ja metodit.



MainWindow – Pitää huolta ikkunan luomisesta, hallitsee eri näkymien välillä liikkumista.

save_load – Sisältää funktiot tallentamiseen ja lataamiseen.

MainMenu – Yksinkertainen aloitusruutu.

Startmenu – Käyttöliittymä uuden pelin luomiseen.

PlayerInfoField – Pelaajan tietojen syöttämiseen. Näitä näytetään StartMenussa pelaajamäärän verran.

PlayWidget – Käyttöliittymä itse pelin pelaamiseen. Pitää huolen, että ruudulla näkyy pelin todellinen tilanne.

CardView – Esittää pöydällä olevat pelikortit graafisesti. Sallii korttien valinnan klikkaamalla. Näyttää myös tekoälyn siirrot.

HandCardView – Esittää kädessä olevat kortit. Erona CardView:hin on, että vain yhden kortin voi valita kerallaan.

CardGraphicsItem – Visuaalinen representaatio pelikortista. Kortista voi näyttää selkäpuolen tai kuvapuolen.

PlayerInfo – Näyttää pelaajien tilastoja. Helppo lisätä tai poistaa pelaajamäärän mukaan.

Game – Itse pelin toteutus. Hoitaa siirtojen toteuttamisen, pistelaskun, korttien jakamisen jne.

Player/AIPlayer – Luokka pelaaja-olion esittämiseen. Pitää kirjaa pisteistä ja pelaajan korteista.

validator – Sisältää funktiot siirron laillisuuden tarkistamiseen sekä tekoälyn siirron määrittämiseen.

brain – Sisältää kirjaston tekoälyn eri vaikeusasteiden kertoimista. Kertoimet vaikuttavat siirron mielekkyyden pisteytykseen.

Useimmat luokat perivät joltain Qt:n luokalta, jotta saadaan käyttöön graafisia widgettejä ja voidaan toteuttaa luokkien välistä kommunikaatiota signaaleilla.

Algoritmit

Tarkistusalgoritmi: is_valid_combo(hand_card, table_cards) - Vertailuarvoksi annetaan pelaajan valitseman käsikortin arvo kädessä. Tätä verrataan listaan, jossa on pelaajan valitsemien pöytäkorttien arvot pöydässä. Jos mikään listan arvoista ylittää vertailuarvon, testi epäonnistuu. Saman suuruiset arvot poistetaan listasta. Seuraavaksi listan arvot lasketaan yhteen. Jos summa ei ole jokin vertailuarvon monikerta, testi epäonnistuu. Näissä yksinkertaisissa testeissä voidaan napata ilmeiset laittomat siirrot (ja näin nopeuttaa suoritusta), sekä saadaan mahdollisesti supistettua listan kokoa. Tämän jälkeen siirrytään raskaampaa laskentaa vaativaan vaiheeseen. Vertailun mahdollistamiseksi lista järjestetään. Nyt etsitään listasta kaikki mahdolliset kombinaatiot, joiden summa on vertailuarvo. Kombinaatiot luodaan itertoolsin combinations-funktion avulla. Kombinaatioista tehdään lista ja sitten tarkistetaan, jos yhden tai useamman listan yhdistelmä tuottaa listan, joka on samanlainen kuin alkuperäinen lista. Jos tällaista yhdistelmää ei löydy, testi epäonnistuu. Muussa tapauksessa testi menee läpi.

AI:n siirron määrittely: get_move(hand_cards, table_cards) – Funktio saa AI-pelaajan käsikortit ja pelin pöytäkortit.

AI:n toiminnan periaatteet:

muodosta comboja iteroimalla -> testaa onko hyväksytty -> pisteytä -> jos korkeimmat pisteet tähän mennessä, säilytä combo -> kun kaikki käyty läpi, tee siirto säilytetyllä comballa -> jos ei ole siirtoa, päätä, mikä kortti asetetaan pöytään -> päätös voidaan tehdä esim. valitsemalla arvoltaan pienin, tai seuraavasti -> laita yksi kortti kerrallan pöydälle ja katso miten hyviä comboja muilla korteilla nyt voisi nostaa -> valitse kortti, jolla paras mahdollinen nosto ensi vuorolla -> erikoiskorttien asettamisesta pöytään miinuspisteitä

Siirtojen pisteytys perustuu kertoimiin, jotka määräytyvät vaikeustason mukaan. Helpommat vaikeustasot suosivat huonompia siirtoja.

Pelissä ::--> kutsu get_move -funktiota -> funktio palauttaa siirron kortteina -> nämä kortit asetetaan valituiksi -> suoritetaan siirto normaalisti

Pistelasku: count_points() – Arvokorteista ja mökeistä lisätään suoraviivaisesti pisteet kullekin pelaajalle. Korttien ja patojen määrää verrataan, ja eniten kutakin kerännyt saa pisteet. Tasatilanne merkitään muistiin, jolloin pisteet siirtyvät seuraavalle kierrokselle.

Vaihtoehtoisia lähestymistapoja olisi esimerkiksi tarkistusalgoritmissa voinut olla rekursiivinen toteutus. Tällöin suurilla korttijonoilla muistinkäyttö olisi mahdollisesti voinut olla pienempi. Toteutus tuntui olevan turhan hankalaa, eikä todellisesta hyödystä ollut varmuutta. Pelissä myös harvoin esiintyy suuria määriä pöytäkortteja kerrallaan, joten uskon toteutukseni olevan täysin riittävä.

Tietorakenteet

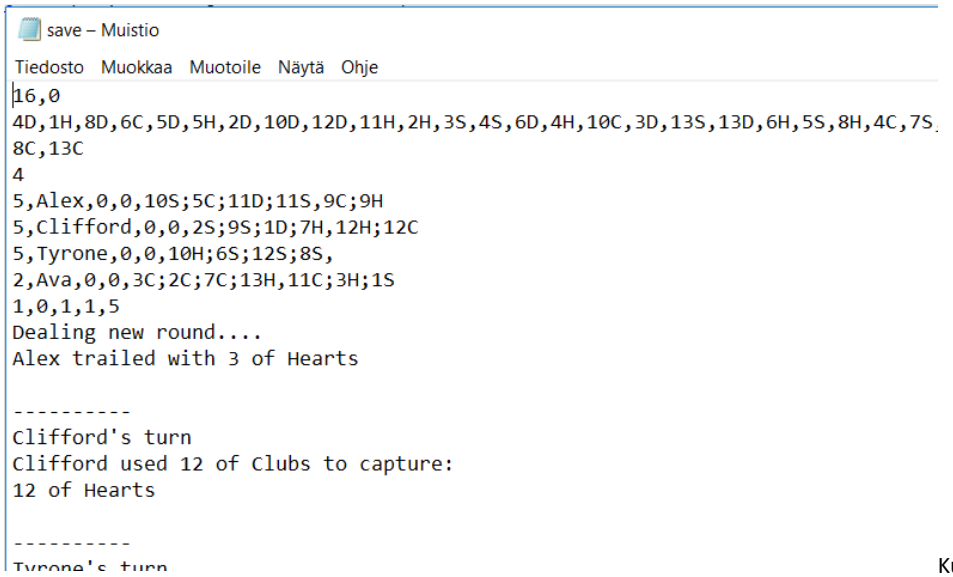
Ohjelmassa on käytetty vain muuttuvatilaisia tietorakenteita. Tärkein näistä on lista, jota käytetään monessa kohtaa pelaajien jo korttien ryhmittämiseen. Listat ovat hyvin kätevä tähän tarkoitukseen, sillä niitä on helppo iteroida ja muokata.

Joissain kohdissa on hyödynnetty myös kirjastoja. Tallennuksessa, latauksessa ja pelikorttien kuvakkeiden latauksessa korttien maiden nimet on kätevä muuttaa tarvittavaan muotoon. Kirjastoa on käytetty myös tekoälyn kertoimien varastointiin. Näin on saavutettu siisti, helposti luettava ja helposti muokattava esitys.

Tiedostot

Pelikorttien ja käyttöliittymän kuvakkeet ladataan PNG-tiedostoista. Kaikki kuvatiedostot löytyvät img-kansiosta. Ääniefektit ovat wav-formaatissa, taustamusiikki on mp3-muodossa. Nämä löytyvät sound-kansiosta.

Itse luomani tiedostoformaatti on pelin tallennus. Se on tekstitiedosto, joka sisältää kaiken tarvittavan tiedon pelitilanteen uudelleen luomiseen.



Kuvakaappaus tallennetusta pelistä

Tiedosto muistuttaa monilta osin csv-formaattia. Tiedot on jäsennelty seuraavasti:

1. rivi – pisteraja, pelimoodi(pakkakasino = 0, perinteinen = 1)

2. rivi – pakassa olevat kortit pilkulla erotettuna

3. rivi – pöydässä olevat kortit pilkulla erotettuna

4. rivi – pelaajien määrä i

seuraavat i riviä:

pelaajan tiedot – vaikeustaso (0 – 4 =AI, 5=ihminen), nimi, pisteet, mökit, käsikortit puolipisteellä erotettuna, nostetut kortit

(5+i). rivi – nykyisen pelaajan indeksi, viimeksi nostaneen indeksi, korttitasapelin kerroin, patatasapelin kerroin, kierroksella tehtyjen siirtojen määrä

lopputiedosto – lokin teksti

Testaus

Alkuperäisistä suunnitelmista:

Tarkistusalgoritmia voi testata syöttämällä sille erilaisia arvolistoja ja tarkistamalla, että ne menevät tai eivät mene läpi asianmukaisesti. Tämä on helposti toteutettavissa Pythonin yksikkötesteillä. Testausta helpottaa se, että algoritmille riittää syöttää lista numeroita, eikä esimerkiksi pelikortteja tarvitse luoda.
Tallennetun pelin latausta pitää myös testata. Ohjelman tulee ilmoittaa virheellisestä tiedostosta ja sen pitää osata luoda pelitilanne oikein.
Lisäksi on testattava, että tekoäly tekee järkeviä siirtoja erilaisissa tilanteissa. ….. Tallennusta täytyy myös testata. Tallennusta ja latausta voi kokeilla erilaisissa pelitilanteissa, ja varmistaa, että kaikki toimii asianmukaisesti.

Ohjelman testaus toteutettiin pääasiassa käyttämällä ohjelmaa. Virheellistä toimintaa havaitessani paikansin, missä kohdassa koodia virhe ilmenee. Virheen juurisyy ja ratkaisu löytyi usein print-komentoja ja Eclipsen debuggeria hyödyntämällä. Joskus taas nettihaut ja Qt:n dokumentointi toivat ratkaisun ongelmaan. Uusia ominaisuuksia luodessani testasin niitä usein myös erillään muusta ohjelmasta, ja vasta toimivuuden varmistuttua integroin ominaisuudet ohjelmaan. Tämän jälkeen testasin vielä integroinnin toimivuutta. Saatuani tekoälyn toimimaan laitoin tietokonevastustajat pelaamaan toisiaan vastaan, jolloin ohjelma myös ikään kuin testasi itse itseään, eikä minun itse tarvinnut olla koko ajan antamassa syötteitä. Tallennusominaisuus valmistui melko myöhään, joten sen testaus jäi hieman puutteelliseksi.

Tarkistusalgoritmi valmistui aikaisessa vaiheessa projektia, ja loin sille jo varhain yksikkötestit, mitkä se läpäisi. Myöhemmin algoritmi muuttui hieman (ottaen syötteeksi kortteja numeroiden sijaan) joten testipenkkiä piti hieman muokata. Yksikkötestejä olisi voinut tehdä myös luokkien metodeille, mutta tähän aika ei riittänyt.

3 parasta ja 3 heikointa kohtaa

+Käyttömukavuus – Ohjelma on suunniteltu mahdollisimman intuitiiviseksi ja helpoksi käyttää. Paljon toimintoja on automatisoitu ja käyttäjältä vaaditaan syötettä vain olennaisissa kohdissa (Esimerkiksi: jos pelissä on vain yksi ihmispelaaja, hänen kortit näkyvät ilman show hand -napin painamista, autofill-toiminto pelin aloituksessa). Virhetilanteita on hyvin vaikea saada aikaan (pl. tallennus ja lopetus).

+Luokkajako – Luokkajako on mielestäni järkevä, ja toiminnallisuudet on jaettu siten, että ne on suhteellisen helppo löytää ja ohjelman laajentaminen ei ole kovin vaivalloista. Olen pyrkinyt luomaan luokat asioista, joita tarvitsee monistaa paljon (esim. Card, PlayerInfo, PlayerInfoField), sekä asioista, joiden kanssa on helpoin vuorovaikuttaa metodien avulla (esim. Game, CardView).

+Tehokkuus ja tekoäly – Algoritmit on pyritty saamaan toimimaan mahdollisimman tehokkaasti. Tekoäly vaikuttaa tuottavan järkeviä siirtoja, ja tarkistusalgoritmi toimii aina. List comprehensionia olen käyttänyt aina kun se on järkevää, ja toisteista koodia olen välttänyt luomalla funktioita. Näin myös koodin luettavuus on parantunut.

-Visuaalinen ilme – Peli näyttää vielä paikka paikoin melko karulta ja alkeelliselta. Ilmettä voisi melko helposti kohottaa lisäämällä kuvakkeita ja tekstuureita, ja esimerkiksi osan painikkeista voisi näyttää kuvakkeiden avulla.

-Kommenttien vähyys – Koodista saisi ymmärrettävämpää runsaammalla kommentoinnilla.

-Pelin lopetus – Ks. puutteet ja viat

Poikkeamat suunnitelmasta

Ominaisuuksia kehittäessä luokkia syntyi suunniteltua enemmän. Tämä oli kylläkin odotettavissa, sillä usein vasta toteutusvaiheessa havaitsee tarpeen uusille luokille. Uudet luokat vähensivät koodin toisteisuutta ja estivät muiden luokkien paisumisen valtaviksi.

Aikataulu myös muokkaantui hieman. Tekoälyn kehitys siirtyi aikaisemmaksi. Tästä oli lopulta suuri hyöty, kun peliä pystyi testaamaan antamalla konevastustajien hoitaa pelaamisen. Tallennusominaisuuden teko taas siirtyi aivan loppuvaiheille.

Aikaa kului huomattavasti enemmän kuin suunnitelmassa. Koodin kirjoittamiseen kului ehkä suunniteltu noin 150 tuntia, mutta ainakin saman verran aikaa kului testaamiseen, tiedonhakuun ja oikeiden Qt:n komponenttien löytämiseen. Monessa tilanteessa ei ollut heti ilmeistä, mitä Widgettiä tms. halutun ominaisuuden toteuttamiseen kannattaisi käyttää. Loppua kohti tämä kuitenkin helpottui, kun Qt:n ominaisuudet kävivät tutummiksi.

Toteutunut työjärjestys ja aikataulu

<u>Helmikuu</u>
Tarkistusalgoritmi: 15h
Perusluokkien toteuttaminen (Game, Card, Player…) ja niiden testaaminen tekstipohjaisesti: 20h
Yksikkötestaus ja debuggaaminen: 20h
Graafisten ominaisuuksien kokeilut ja suunnittelun alku (erillään pelistä): 40h

<u>Maaliskuu</u>
Ei paljon muutoksia, pienempää näpertelyä mm. CardView:hin liittyen n. 20h

<u>Huhtikuu</u>
Luokkajaon päivittäminen: 10h
Pelin integroiminen graafiseen käyttöliittymään: 40h
Tekoälyn algoritmit ja toteutus pelitilanteeseen: 20h
StartMenun toteutus : 15h
PlayerInfon toteutus ja ääniefektit: 15h
Tallennuksen toteutus: 10h
Dokumentointi: 8h

Viitteet

Ainakin stackoverflow ja stackexchange ovat olleet useaan otteeseen tietolähteenä.

Korttipakka png-kuvina: http://acbl.mybigcommerce.com/52-playing-cards/

Taustamusiikin lataamiseen: https://www.mp3juices.cc/

PyQt5:n opiskeluun: http://zetcode.com/gui/pyqt5/

Qt:n opiskeluun: http://doc.qt.io/qt-5/index.html

Esimerkki Kasinon kaltaisesta pelistä: http://www.spigo.co.uk/games/cassino

Ääniefektit: https://opengameart.org/content/card-game-sounds

Lisäksi olen käyttänyt Qt Designeria erilaisiin kokeiluihin ja ottanut mallia ulostulevasta koodista, soveltaen sitä omaan koodiini.

Liitteet

Ohessa vielä muutama kuvakaappaus ohjelmasta.

Starting new game