Ohjelmoinnin harjoitustyö

Bussikuskipeli

Jasmin Kaseva | Ohjelmoinnin perusteet | 1/2024

Pelin kuvaus

Työn aiheena on korttipeli, jossa peli etenee kierroksittain ja jokainen pelaaja pelaa omalla vuorollaan. Tavoitteena on tehdä yksinkertainen arvauspeli, jossa on (juoma)sakkoja vääristä arvauksista. Jokaisella kierroksella panokset kovenevat ja sakkoja tulee enemmän. Kyseessä on siis täysin hyväntuulinen tuuripeli, jonka pitäisi toimia ihan hyvin myös digitaalisena versiona.

Peliä voi pelata yksin, tai maksimissaan 17 pelajaa (3 korttia täytyy riittää jokaiselle). Pelissä käytetään tavallista korttipakkaa, joka sekoitetaan pelin alussa. Ensimmäisellä kierroksella pelaajat arvaavat vuorotellen heille jaettavan kortin värin, eli kyseessä on 50-50 todennäköisyys joutua ottamaan yhden hörpyn. Jaettu kortti jää aina pelaajille talteen.

Toisella kierroksella pelaajien täytyy arvata onko seuraavaksi jaettava kortti suurempi, pienempi vai sama kuin edellisellä kierroksella jaettu kortti, eli jos pelaajalle olisi vaikka jaettu risti 2 aiemmalla kierroksella, hän arvailee sen perusteella. Tässä voi jo hieman koittaa päätellä todennäköisyyksiä, mutta riskinä on kahden hörpyn sakko.

Kolmannella kierroksella arvataan jaettavan kortin maa, ja riskinä on kolmen hörpyn sakko. Peli etenee sitä mukaa paljonko pelaajia on mukana, ja kuinka kauan arvauksissa kestää. Neljännellä ja viimeisellä kierroksella annetaan bonushörppyjä pelaajalle jonka korttien summa on suurin. Eli pelaaja josta tuli niin sanottu bussikuski. Nimi bussikuski ei siis oikestaan tarkoita muuta kuin epäonnista pelaajaa. Oikeassa pelissä on pyramidiarvaus lopussa, mutta se jääköön pois tästä versiosta.

Pelissä ei ole voittajia eikä häviäjiä, ellei itse yritä vältellä sakkoja tai saada niitä. Peliä voi pelata niin kauan kuin haluaa, ja jokaiselle kierrokselle voi tulla uudet pelaajat pelaamaan. Kun pelaaminen lopetetaan tallennetaan tiedot, montako pelaajaa pelasi parhaimmillaan, mikä oli paras ja mikä huonoin tulos, ja montako sakkoa jaettiin kaiken kaikkiaan.

Analysointi

Peli tulostaa heti aluksi tervetuloviestin, ja kertoo lyhyesti mistä on kyse. Seuraavaksi peli pyytää lisäämään pelaajia, jotka tallenetaan peliin. Koska pakassa on 52 korttia, ja jokaiselle pitää pystyä jakamaan 3 korttia, on 17 maksimimäärä pelaajia.

Kyseessä on korttipeli, joten seuraavaksi luodaan tavallinen korttipakka (4 maata ja numerot 1-13). Pakka sekoitetaan heti pelin aluksi, koska muuten pakan päältä arvailu olisi ainakin toivottavasti turhan helppoa. Pelin alussa ei suoraan sanota että kierroksia on 4 ja mitä kierrokislla tehdään, vaan kierrosten säännöt selviää aina seuraavalle kierrokselle mentäessä.

Peli itsessään on todella yksinkertainen, ja tulosteet pyritään pitämään myös tarpeeksi lyhyinä ja selkeinä. Peli ei vaadi oikestaan minkään näköisiä lukuja syötteessä, joten erroritkin pystyy suhteellisen helposti välttämään. Syötteen antamista ohjataan virheviesteillä, käytännössä ainoastaan nimi voi olla mikä tahansa paitsi tyhjä syöte. Ehkä jokin maximi merkkimäärä saattaisi olla tarpeellinen.

Ainakin omasta mielestäni tämänkaltainen peli vaatii erilaisia tulosteita eri tilanteisiin. Esimerkiksi jos jokaisella kierroksella saa sakkoa niin peli tulostaa jonkun herjan, tai jos ei saa lainkaan sakkoa niin onnittelee hyvästä arvailusta jne. Ei ole kovin haastavaa sinänsä lisätä näitä tulosteita peliin mukaan, mutta kertyy heti jonkin verran ylimääräistä sisältöä.

Työn haastavin osuus on ollut selvittää mitä tietoa kannattaisi kirjoittaa tiedostoon ja miten. Tästä johtuen peli pitää tilastoa yksinkertaisesti pelattujen pelien lukumäärästä ja milloin peliä on viimeksi pelattu.

Pelin koodaus

Pelissä on yhteensä 13 funktiota jotka on listattu aakkosjärjestyksessä alle.

aloitaPeli(pelaajat, korttipakka)

Tämä funktio on tehty yhden peli pelaamiseen, eli sitä kutsutaan main() funktion sisällä aina, kun pelaaja on päättänyt jatkaa peliä ja lisännyt ainakin yhden pelaajan. Funktiossa käytetään parametreinä pelaajat sanakirjaa ja korttipakka listaa. Funktion avulla varmistetaan myös että peliin on lisätty vähintään yksi pelaaja ennen ensimmäiselle kierrokselle siirtymistä.

Funktion sisällä on kierrokset sanakirjaan määritelty 0-3 välillä kierrokset. Eli ensimmäinen kierros on 0 ja niin kauan kuin kierros < 4 mennään kierrokset järjestyksessä läpi. Tästä kutsutaan siis järjestyksessä jokainen kierrosfunktio, ja kun kierrokset on pelattu, palataan main() funktioon jossa pelaajat päättävät jatkavatko peliä vai eivät.

ensimmainen_kierros(pelaajat, korttipakka)

Kierrokset ovat rakennettu pitkälti samalla logiikalla keskenään, mutta arvaukset ovat täysin erilaisia, sakkomäärät ovat erilaiset ja tulosteet ovat erilaiset. Ensin varmistetaan että on pelaajia ja korttipakka (ei pitäisi olla mahdollista edes päästä kierrokselle ilman niitä). Kierros mennään for-loopilla pelaaja kerrallaan läpi. Eli pelaajalle tulostuu nimellinen viesti joka kertoo, että on oma vuoro, ja pyytää arvaamaan kortin värin.

Itse arvailu toimii siten, että syötteeseen sopii tarkalleen vain ”musta” tai ”punainen”, (kirjainkoolla ei ole väliä) muuten tulostuu ”virheellinen syöte, valitse musta tai punainen”. Seuraavaksi määritellään että oikea väri on musta jos korttipakan päällimmäinen kortti (korttipakka[0][0]) löytyy listalta jossa on on risti ja pata. Muuten kortti on punainen. Sen jälkeen kortti otetaan pois pakasta ja kortti lisätään pelaajan tietoihin. Nyt seuraavalla pelaajalla on omalla kohdallaan eri kortti jaossa.

Peli antaan tulosteen sen perusteella menikö arvaus oikein vai väärin, ja kun kaikki pelaajat on käyty läpi, peli tulostaa viestin että kierros on päättynyt. Tulosteet ovat samankaltaisia eri kierroksilla, ja f lausetta hyödyntämällä voidaan pelaajalle tulostaa jaettu kortti näkyviin. Tämä toimii jokaisella kierroksella.

kolmas_kierros(pelaajat, korttipakka)

Kolmannella kierroksella on yksinkertaista koodia, eli pelaajan täytyy arvata seuraavan kortin maa, ja vaihtoehdot on listattu numeroineen, joista sitten arvataan/syötetään joku numero. Esimerkiksi jos pata = 1 niin 1 syöttämällä arvaa padan jne. Ohjelma määrittelee oikean maan jaetusta kortista. Peli periaatteessa jakaa kortin ennen kuin se kerrotaan pelaajalle. Eli jos arvaus on sama kuin oikea maa, pelaaja välttyy sakolta. Jos ei ole oikein niin pelaaja saa triplasakot.

lisaaPelaajat()

Arvelin että pelaaja täytyy tallentaa sanakirjaan, jotta sen sisältöä voi muokata ja sinne voi lisätä kortteja. Pelaajien lisääminen onnistuu funktion lisaaPelaajat() avulla. Funktio kysyy aluksi haluatko lisätä pelaajan ja jos vastaat kyllä, funktio pyytää kirjoittamaan nimesi. Jos vastaat ei niin

peli alkaa niillä pelaajilla, jotka peliin on lisätty. Jos ei ole lisätty yhtään pelaajaa peli kysyy haluatko pelata peliä.

lue_peli_lkm()

Funktiota käytetään pelitietojen lukemiseen tiedostosta. Funktion avulla avataan peli_lkm.txt tiedosto ja luetaan sen sisältö. Sieltä löytyy tietoa, kuinka monta kertaa peliä on kaiken kaikkiaan pelattu, ja milloin tiedot ovat viimeksi päivittyneet. Tässä vaiheessa yritin laittaa tietoa sakoista myös, mutta se osoittautui hieman liian työlääksi tähän projektiin. Tiedosto avataan lukumuotoon, sisältö luetaan ja sieltä etsitään tallennettu lukumäärä pelejä, jotta se voidaan lisätä kyseisen pelikerran määrään. Funktio palauttaa pelilukumäärän ja sisällön. Jos mitään ei löydy, funtkio palauttaa 0 ja tyhjän merkkijonon.

luoKorttipakka()

Hyödynsin pelissä oppimaan pakanluontia ja sekoitusta, sekä pakasta nostoa pelin kulkua varten. Eli luodaan lista korttipakka ja määritellään numerot rangella 1-14, ja maat (pata, ruut, risti, hertta). For loopilla luodaan tupleja eli kortteja jotka on muotoa (maa, numero). Random on tuotu peliin pakan sekoitusta varten, ja random.shuffle avulla saadaa pelivalmis korttipakka. Koska peliä pystyy pelaamaan monta kertaa, korttipakka luodaan funktion luoKorttipakka() avula.

main()

Mainfunktiossa on while loop ja siellä on sisällä pelin muita funkiota (kierrokset, pakanluonti jne). Itse en olisi keksinyt tehdä erikseen mainfunktiota ja kutsua sitä, mutta sain pelin testaajalta kehoituksen tehdä siitäkin erillinen funktio jota kutsutaan. Peli lähtee sillä oletuksella että pelataan, joten aluksi pelaa = 1 (1 eli kyllä) ja niin kauan kun pelaa ei ole 0 pelataan uusi peli. Eli aina kun yksi peli on pelattu peli kysyy "Jatketaanko peliä?\n1 Jatka\no Lopeta", muu kuin 1 tai 0 kertoo että syöte ei ole kelvollinen.

Käytännössä peli koostuu täysin funktioista, jotka kutsuvat toisiaan annetussa järjestyksessä/olosuhteissa pelaajan syötteen perusteella. Funktiossa tulostetaan ensin tervetuloviesti, ja tämä viesti tulostuu vain kerran ohjelman ajon aikana, muut toiminnot käydään läpi niin kauan kuin pelaajat jaksavat peliä pelata.

neljas_kierros(pelaajat, korttipakka)

Neljäs eli viimeinen kierros, on sellainen, jossa täysin hyvällä logiikalla se, jonka korttien summa on suurin, saan 4 bonus hörppyä. Tämä tulostetaan pelaajille ja pelaajille tulostetaan myös nimet ja korttien summat. Korttien summaamiseen on käytetty sum metodia joka summa for loopissa läpi käydyt korttien numerot eli kortin indeksissä 1 olleet arvot.

Suurin summa saadaan for loopilla käymällä pelaajat läpi laskemalla summa ja jos summa on suurempi kuin määritelty suurin summa (aluksi 0), niin siitä tulee uusi suurin summa. Kun pelaajat ja summat on tulostettu peli kertoo mikä summa oli suurin, ja pelaaja joka kyseisen summa sai, kuittaa hörpyt.

pelitilanne(pelaajat)

Pelitilanne tulostaa nimensä mukaisesti pelitilanteen, eli pelaajat, heidän kortit sekä sakot. Tulosteen muotoiluun on hyödynnetty for-loopia, joka käy pelaajat yksitellen läpi ja tulostaa f-lausetta hyödyntäen yllä mainitut tiedot. Esimerkiksi kortteja tulostettaessa on hyödynnetty join metodia *print(f" Kortit: {', '.join(map(str, tiedot['kortit']))}").* Tarkoituksena on antaa pelaajille tilannekatsaus, josta saa helposti selvää.

seuraava_kierros()

Tämä funktio on mahdollisesti turha, mutta omasta mielestäni selkeyttää hieman pelin kulkua. Eli funktiota kutsutaan aloitaPeli funktion sisällä ennen siirtymistä seuraavalle kierrokselle. Pelaajalle tulostuu viesti ”paina enteriä jatkaaksesi” ja enteriä painamalle siirrytään eteenpäin. Jos paina jotain muuta, ohjelma tulostaa ”älä kirjoita mitään”. Nyt kierrokset erottaa toisistaan helposti, ja tämä tuo jopa vähän lisää interaktiivisuutta peliin, koska pelaajan täytyy tehdä jotakin että peli etenee.

tallenna_peli_lkm(pelit_lkm)

Pelin tietojen tallentamiseen luotu funktio hyödyntää lue_peli_lkm funktion palauttaamaa aiempien pelien lukumäärää, ja lisää siihen tämänhetkisen pelimäärän. Funktio siis kutsuu lue_peli_lkm funktiota, josta se saa tarvitsemansa arvon. Sitten funktio avaa samannimisen tiedoston (peli_lkm.txt) kirjoitusmuodossa, jolloin aiempi versio tuhoutuu, ja tilalle luodaan uusi tiedosto. Tällä tavoin peli voi automatisoida tiedoston lukemisen ja kirjoittamisen, eikä pelaajan tarvitse tietää tiedostojen nimiä.

toinen_kierros(pelaajat, korttipakka)

Toinen kierros on kaikista monimutkaisin, koska siinä hyödynnetään ensimmäisen kierroksen korttia. Eli tätä kierrosta varten on luotu korttien vertailu funktio, joka esitellään tarkemmin seuraavaksi. Pelaajan täytyy arvata onko seuraava kortti suurempi vai pienempi kuin hänelle ensin jaettu kortti. Ensin varmistetaan että on olemassa pelaajat ja korttipakka, ja että pelaajalla on ainakin yksi kortti. Jos tälläinen kortiton pelaaja eksyisi kierrokselle, siirryttäisiin suoraan seuraavaan.

Peli pyytää käyttäjältä syötettä, eli arvausta, ja tulostaa samalla edellisen kierroksen kortin, jokainen ei sitä välttämättä muista. Siiten ohjelma menee eteenpäin ja määrittelee pelaajan ”edellisen kortin” ja jakaa pelaajalle uuden kortin. Nämä kaksi korttia lähetetään vertailtavaksi toiseen funktioon, joka palauttaa numeron. Eli esimerkiksi 1 tarkottaisi että kortti on suurempi kuin edellinen kortti. Jos funktion palauttama numero on yhtä kuin arvaus, pelaaja välttyy sakoilta. Muuten pelaaja saa viestin ottaa 2 sakkohörppyä.

vertaa_kortteja(edellinen_kortti, uusi_kortti)

Korttien vertailua varten luotiin yksinkertainen funktio, joka palauttaa numeron 1-3 sen perusteella kumpi kortti (edellinen vai uusi) oli suurempi tai pienempi tai oliko kortit samat. Tämä on toteutettu yksinkertaisella if-else rakenteella ja korteista vertailla 1 indeksiä, eli numeroa. Maalla ei ole tässä kohtaa mitään merkitystä.

Ulkopuolisia kirjastoja joita ohjelma käyttää ovat random ja datetime. Randomia käytetään korttipakan sekoittamisen tarpeeksi sekaisin, ja datetime käytetään tiedostoon kirjoittamiseen, jotta sinne saa aikaleiman tietojen päivittämisestä.

Peli kertoo selkeästi edetessään mitä käyttäjän täytyy tarvittaessa syöttää. Arvaukset ovat käytännössä monivalintatehtäviä, joten virhemarginaali käyttäjän puolelta on aika pieni, jos syöte ei kelpaa. niin peli odottaa niin kauan että käyttäjä syöttää sopivan syötteen ennen kuin peli jatka eteenpäin.