Vaakapeli loppuraportti

Jonatan Lygdman 540 654

Informaatioverkostot, 1. vuosikurssi, 26.4.2017.

Olen toteuttanut vaakapelin keskivaikealla vaikeustasolla. Vaakapelin ideana on asettaa punnuksia peliin ilmestyville vaaoille, sillä ehdolla, että kaikki pelin vaa'at pysyvät jokaisen siirron jälkeen tasapainossa. Projektin toteutuksessa pyrin mahdollisimman hyvin seuraamaan ohjeistusta. Pelissä vaa'at pysyvät tasapainossa, mikäli vasemman ja oikean puolen epätasapaino on enimmillään yhtä kuin vaa'an koko (pelissä vaakoja on kokoa 3, 5, 7 ja 9) jaettuna kahdella pyöristettynä alaspäin. Toisin sanoen suurin sallittu epätasapaino on sama kuin se, että asettaisi yhden punnuksen vaa'alla mahdollisimman kauas tuki- eli keskipisteestä. Mikäli asetettu punnus saisi minkä tahansa pelin vaaoista epätasapainoon, se menetetään ja vuoro siirtyy seuraavalle pelaajalle. Pisteet punnuksista kertautuvat jokaisen vaa'an alla olevan vaa'an mukaan, joten mitä korkeammalle vaa'an asettaa, sitä enemmän pisteitä keräät! Punnuksia voi myös laittaa toisten päälle, jolloin saa kaikki kyseisessä paikassa olevat punnukset haltuunsa.

Ohjelmaa käynnistettäessä käyttäjä näkee ensin dialogin. Tekstikentän tyhjäksi jättäminen aloittaa pelin niillä pelaajilla, joiden nimet on siihen mennessä dialogeihin syötetty. Kun halutut pelaajat on syötetty, peli voi alkaa. Tässä vaiheessa näytölle aukeaa itse pelinäkymä. Pelinäkymä on pelin alussa aina samanlainen, alas keskelle piirtyy pelin ”initialScale”, eli pelin aloittava vaaka, joka on joka kerta samanlainen. Peliä ohjataan pelialuetta klikkaamalla. Klikkaamalla hieman vaakojen yläpuolelta, asettuu punnus klikattuun paikkaan. Jos paikassa on jo punnus, voi sen päälle asettaa toisen punnuksen klikkaamalla jo olemassa olevaa punnusta. Punnuksen päällä oleva numero kertoo, kuinka monta punnusta kohdassa on. Pelin valikosta löytyy kohta ”Apua!”, jossa on hyvin lyhyesti selitetty samat asiat kuin tässä.

Ohjelmakoodi on jaettu kuuteen luokkaan: Game, Player, Scale, GameGUI, Weight sekä StackableObject, joista viimeinen on abstrakti luokka jonka aliluokkia Weight ja Scale ovat. Tämä mahdollistaa molempien tyyppien lisäämisen samoihin buffereihin joka on tarpeellista, kun pidetään huolta siitä, mitä vaa'an päällä on. Luokista suurimmat ja monimutkaisimmat ovat Scale ja GameGUI. Scale-luokka pitää huolen koko ohjelman oikean toiminnan kannalta tärkeimmistä metodeista, nimittäin vaakojen tasapainon tarkistaminen sekä vaakoihin punnusten ja toisten vaakojen lisääminen. Metodit isBalanced, addScale ja addWeight ovat näistä vastuussa. Myös pistelaskumetodi countPoints on toteutettu Scale-luokassa, jotta sitä voidaan GUI:ssa kutsua pelin alkuperäiselle vaa'alle, sillä tämä metodi käy rekursiivisesti läpi kaikki pelin vaa'at. GUI-luokassa on muutamia keskeisiä metodeja, erityisesti drawScale joka piirtää vaa'an peliruudulle. Muut neljä luokkaa ovat melko yksinkertaisia ja lähinnä tukevat näiden kahden toimintaa.

Haastetta luokkien väliselle kommunikaatiolle loi esimerkiksi se, että tietty vaaka tietää vain vaa'at jotka ovat sen yläpuolella, mutta ei alapuolella. Tämän takia koin hyväksi ratkaisuksi, että joitakin metodeita kutsutaan pelin alkuperäiselle vaa'alle, sillä se varmistaa sen, että kaikki pelin vaa'at tulee käytyä läpi. Vaaka ei siis tiedä missä pelissä se on, ainoastaan pitää huolen punnuksista ja muista vaaoista jotka ovat sen yläpuolella.

Pyrin pitämään luokkien riippuvuuden toisistaan mahdollisimman pienenä. Game-luokka yhdistää nämä yksinkertaisella tavalla luomalla pelin, jossa on haluttu määrä pelaajia sekä pelin ensimmäinen vaaka. Myös uusien vaakojen luominen on tämän luokan vastuulla. Alkuvaiheen liika luokkien riippumattomuus johti kuitenkin siihen, että mielestäni GUI-luokka ottaa pelin ohjaamisesta ja ylläpitämisestä liikaa vastuuta, kun yritin sen avulla sitoa pelin eri osia yhteen. Tämä ei ole paras mahdollinen ratkaisu, vaan metodeja tulisi mielestäni mieluummin sijoittaa enemmän eri luokkiin. Myös punnusten sekä vaakojen asettamismetodit olisivat voineet olla paremmin kytköksissä niitä vastaavien piirtometodien kanssa. Nyt ne ovat melko irrallaan toisistaan, ja ne voisi varmasti toteuttaa niin, että molemmat asiat hoituisivat samalla metodikutsulla.

Ohjelmassa on käytetty keskeisimpien ongelmien ratkaisemiseksi rekursiivisia algoritmeja. Sekä metodit totalWeight että isBalanced, kaksi ohjelman keskeisimmistä metodeista, on toteutettu rekursiivisesti. Niiden toimintaperiaate on melko samanlainen. IsBalanced – metodissa olisi voinut hyödyntää totalWeight – metodia apuna, mutta toteutin nämä sitä varten väärässä järjestyksessä, isBalanced ensin. Ne toimivat kuitenkin seuraavan laisella periaatteella: Käy läpi jokainen vaa'an mahdollisista kohdista. Mikäli kohta on tyhjä, älä tee mitään. Mikäli kohdassa on yksi tai useampi punnus, lisää niiden aiheuttama momentti (isBalanced) tai paino (totalWeight) laskuriin. Mikäli kohdassa on vaaka, kutsu samaa metodia rekursiivisesti kyseiselle vaa'alle. Matemaattinen laskenta pelissä hoidetaan siis rekursion kautta. Päädyin tähän ratkaisuun sen takia, että vaakoja saattaa olla pelissä hyvinkin suuri määrä, mutta rekursiolla kaikki vaa'at voidaan käydä tehokkaasti läpi. Myös se, että totalWeight on toteutettu rekursiivisesti, ja jokaiselle punnukselle on annettu painoksi 1, mahdollistaa ohjelmalle jatkokehitysmahdollisuuden, jossa punnuksia voisi olla eri painoisia. Tämän toteuttaminen nykyisellä ohjelmakoodilla ei olisi kovinkaan vaikeaa.

Pelin koodi perustuu suurelta osin muuttuvatilaisiin tietorakenteisiin. Pelin vaakojen tila muuttuu jatkuvasti, jonka takia Arrayt sekä Bufferit olivat mielestäni helpoin ja selkein ratkaisu tilan säilyttämiseksi. Myös var:it osoittautuivat hyödyllisiksi pisteitä laskettaessa sekä esimerkiksi vaa'an kokonaispainoa laskiessa. Arrayt sopivat erityisen hyvin pitämään tallessa tietoja vaa'an päällä olevista objekteista, sillä vaakojen koko ei muutu, mutta niiden päälle voidaan aina asettaa lisää objekteja.

Ohjelman testaus asetti välillä haasteita, sillä esimerkiksi tasapainon määritelmässä käydään läpi kaikki pelin vaa'at ja jos jokin näistä on epätasapainossa, on vaikea tietää mikä. Selvisin kuitenkin ilman suurempia ongelmia aivan yksinkertaisten yksikkötestien sekä println:ien avulla. Testausta en suunnitellut tarpeeksi hyvin, sillä se osoittautui hyvinkin hankalaksi. Testasin ensin pelin mekaniikkoja yhdellä vaa'alla ennen kuin siirryin useampaan vaakaan, mutta rekursiivisten metodien takia ongelmia oli vaikea löytää. Haastavin tapaus oli se, että peli ei toiminut halutulla tavalla, jos asetettu punnus sai jonkin alla olevan vaa'an epätasapainoon, sillä rekursiivinen metodi laski uuden tasapainon ainoastaan sen yläpuolella oleville vaaoille. Jonkin ajan jälkeen ja

monia eri yhdistelmiä kokeilemalla sain kuitenkin ongelman hallintaan. En kuitenkaan tiedä miten tätä olisi voinut järkevästi testata.

Peliin punnuksia asetettaessa on klikkauksen kohta turhan tarkka. Tästä ei pitäisi koitua haittaa, sillä ohjelma ei tee mitään, ellei klikkausta suoriteta sallitulle alueelle, mutta ensimmäistä kertaa pelaavalle se saattaa aiheuttaa hämmennystä. Tämä johtuu siitä, että sekä x- että y-koordinaatin täytyy olla melko spesifien rajojen sisällä. Tämän voisi korjata sillä, että antaisi y-koordinaatille vähän enemmän pelivaraa, tai sillä, että piirtäisi peliruudulle esimerkiksi ulkoreunat alueelle, johon klikkaamalla punnuksen voisi asettaa. Tämä alue on kuitenkin ohjelmassa olemassa, niin sen voisi varmasti ruudulla myös esittää. Pelissä voisi olla hyvä olla myös uusi peli – nappi.

Paras kohta ohjelmassa on ehdottomasti totalWeight – metodi. Se toimii yksinkertaisella rekursiolla ja hyvin lyhyellä koodinpätkällä erinomaisesti juuri haluttuun käyttötarkoitukseen. Se myös antaa tilaa muuttaa yksittäisen punnuksen painoa ja toimisi silti aivan samalla tavalla. Muut rekursiivisesti toteutetut metodit toimivat myös hyvin mutta eivät ole yhtä yksinkertaisella tavalla toimivia kuin tämä.

Heikkona kohtana on pakko mainita tapa, jolla tarkistetaan johtaako punnuksen asettaminen klikattuun kohtaan systeemin epätasapainoon. Koodi lisää ensin punnuksen klikattuun kohtaan, ja jos epätasapaino syntyy, niin punnus poistetaan. Tämä olisi järkevämpää toteuttaa tavalla, jossa ensin tarkistettaisiin johtaako tämä siirto epätasapainoon, ja sitten vasta punnus lisättäisiin. Yritin tehdä metodia tällä tavalla, mutta en saanut sitä millään toimimaan, joten keksin ongelmalle vaihtoehtoisen ja toimivan, mutta ehdottomasti huonomman ratkaisun, sillä se on alttiimpi virheille.

Myös kohdassa, jossa punnus asetetaan pelin GUI:hin on eräs kohta, joka ei itseäni oikein tyydytä. Olen myös kommentissa lyhyesti selostanut tilanteen. Siinä if-lauseen sisällä asetetaan punnus pelin, mikä on mielestäni jokseenkin hämmentävää, sillä itse ainakin pyrin if-lauseilla ainoastaan tarkistaa totuusarvoja enkä suorittaa tilaa muuttavaa koodia. Kirjoitin pätkän tuolla tavalla, sillä se oli yksinkertaisin tapa saada koodi toimimaan oikein, mutta koodityyliltään se ei ole paras mahdollinen.

Tein kaiken melko lailla samalla tavalla kuin olin suunnitellut. Lisäsin ”pinottaville” objekteille, eli vaaoille ja punnuksille, oman abstraktin yliluokan jota en ollut suunnitellut, mutta muuten toimin pitkälti suunnitelman mukaisesti. Kokonaisuudessaan suunnittelemani ajankäyttöarvio oli melko hyvä. Sen sisältämien osien ajankäyttö ei. Perusominaisuudet päätin pitää hyvin yksinkertaisina ja näin ollen asioiden toisiinsa sitomiseen meni enemmän aikaa kuin olin suunnitellut. Graafisen käyttöliittymän toimintoihin meni enemmän aikaa kuin olin suunnitellut, kun taas toimintoihin vähemmän. Suunnitelmasta poikkesi myös se, että toteutin käyttöliittymän jo hyvin aikaisessa vaiheessa. Päädyin tähän ratkaisuun kun huomasin, että testatessa ilman minkäänlaista visuaalista apua oli koodia ja sen toimintaa melko vaikea ymmärtää. Käyttöliittymä helpotti ymmärtämään ongelmia uudella tasolla.

Pelin lopputulokseen olen melko tyytyväinen. Se on järkevä kokonaisuus, jossa kullakin osalla on oma roolinsa suuremmassa kokonaisuudessa. GUI-objektista olisi voinut jakaa vastuuta muille luokille melko paljonkin, sillä se on liian keskeisessä osassa ohjelman toiminnan kannalta. Myös se, että esimerkiksi piirto on monessa kohtaa erillinen funktio itse punnuksen tai vaa’an asettamisesta hankaloittaisi jatkoparannuksia. Olen tyytyväinen panostukseeni projektia varten ja lopputulos on mielestäni onnistunut. Työ oli vaikeampi toteuttaa kuin aluksi olisin uskonut, mikä tekee lopputuloksen tarkastelemisesta tyydyttävää. Peli on pelattavuudeltaan kiehtova idea, joka toimii parhaiten kolmella tai neljällä pelaajalla pelattaessa. Pelissä olisi potentiaalia netissä pelattavalle moninpelille sen sijaan, että neljä ihmistä istuisivat saman tietokoneen ääressä. Pelin pelattavuuden kannalta on mielestäni myös hieman epäreilua ja kohtuutonta asettaa uusi vaaka jokaisen pelatun kierroksen jälkeen. Asia oli kuitenkin näin ohjeistettu, joten toteutin sen näin. Pelattavuuden kannalta parempi olisi, että esimerkiksi viiden asetetun punnuksen jälkeen syntyisi uusi vaaka, pelaajien lukumäärästä riippumatta. Koska uusi vaaka syntyy jokaisen kierroksen jälkeen, syntyy niitä kaksinpelissä turhan paljon. Myös neljällä pelaajalla pelattaessa on ensimmäisellä pelaajalla etulyöntiasema, sillä uusi vaaka syntyy aina ennen hänen vuoroansa. Jatkokehityksessä miettisin tälle mekaniikalle järkevimmän toteutuksen.

Jatkokehitettävää jäi myös pelin ulkonäölle, erityisesti piste- sekä vuoroindikaattoreille. Vaakojen ja punnusten ulkonäkö on mielestäni tyydyttävä.

Apua minua askarruttaneisiin ongelmiin olen saanut StackOverflow-sivustolta (http://stackoverflow.com/), sekä vanhemmilta informaatioverkostojen tieteenharjoittajilta.