Patrik Ahvenainen / 013326292
patrik.ahvenainen@helsinki.fi
Karstulantie 8 C 43
00550 Helsinki

Yleinen noppapeli

Toteutusdokumentti

Python 2.4

Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Ohjelmoinnin harjoitustyö (python)
Ohjaaja: Joel Rybicki

Palautuspäivä: 04.05.2010

Sisältö

Käyttöohje

Esittely

Ohjelma on viihdekäyttöön tarkoitettu peli yhdelle tai useammalle ihmispelaajalle. Pelin alussa pelaajat sopivat mitkä yhdistelmät peliin otetaan käyttöön, kuinka monta noppaa pelissä on, mikä on noppien suurin sallittu silmäluku sekä kuinka monta heittoa yhdellä vuorolla voi suorittaa. Pelaaminen tapahtuu joko hiirellä tai näppäimistöllä.

Ajo-ohje

Pelin tiedostot ovat kahdessa paketissa eli kansiossa 'GUI' ja 'Game'. Pelissä käytettävät luokat ovat suurimmaksi osaksi samannimisten moduulien (tiedostojen) sisällä (Taulukko 1). Tiedostorakenteen tulisi olla esitetyn mukainen.

Pelin graafinen käyttöliittymä käynnistetään ajamalla Main.py -tiedosto komennolla

```
python Main.py
```

Komennon ”python” tulisi ajaa ohjelma python-tulkin (version 2.4) läpi.

Taulukko 1. Graafisen käyttöliittymän ajamiseen tarvittavat paketit, moduulit ja luokat ja niiden keskinäinen rakenne.

Paketti	Moduuli	Luokka
Game	Combination	Combination
	Dice	Dice
	Game	Game
	HighScores	HighScores
	Logger	Logger
	Player	Player
	ScoreList	ScoreList
	Settings	Settings
	Throw	Throw
GUI	DiceRow	DiceRow
	GameState	GameState
	GUI	-
	Main	-
	ScoreBoard	ScoreBoard
		ScoreBoardScore
		TotalScore
	SimpleCallback	SimpleCallback

Laitteistovaatimukset

Ohjelma on suunniteltu toimimaan pythonin versiolla 2.4. Toimivuus muilla versioilla on mahdollista, mutta ei taattua. Ohjelman tulee kyetä luomaan kansioita siihen kansioon, mistä sitä ajetaan ja luomaan näihin kansioihin tiedostoja. Tiedostot ovat kooltaan melko pieniä, eikä paljon kovalevytilaa siis tarvita. Ohjelman kokonaiskoko on alle 1 MB. Näytön suositusresoluutio on vähintään 1024x800.

Pelin ajamiseen tarvitaan lisäksi Tkinter-moduulia. Lisäksi käytetään muutamaan Tkinteriin liittyvää moduulia: `tkFileDialog`, `tkMessageBox`, `tkFont`. Lisäksi joissakin ohjelman luokissa käytetään seuraavia moduuleja: `os`, `logging`, `logging.handlers`, `random`, `pickle`, `datetime`, `unittest`, `time`, `sys`.

Nämä moduulit ja python kuuluvat nykyisin monen käyttöjärjestelmän perusasennuksiin. Jos niitä ei ole valmiiksi asennettuina, ne löytyvät Internetistä vapaasti asennettavina useammalle käyttöjärjestelmälle.

Peliä on testattu osittain seuraavilla kokoonpanoilla:

Asus G2Pc kannettava tietokone: ***Näyttö***: 17" WXGA+ (1440 x 900); ***Prosessori***: Intel Core 2 Duo T7200 (2.0Ghz); ***Keskusmuisti***: 2x1GB DDR2; ***Näytönohjain***: ATI Mobility Radeon X1700 512MB; ***Käyttöjärjestelmä***: Ubuntu 9.10; ***Python-versiot***: 2.4 ja 2.6.

Jimm's Pro Gamer GTX 295 pöytäkone: ***Näyttö***: 24" (1920 x 1200); ***Prosessori***: Intel Core i7 920 2.66Ghz; ***Keskusmuisti***: 3x2GB Elite, DDR3 1.3Ghz; ***Näytönohjain***: GeForce GXT 295 1792MB GDDR2; ***Käyttöjärjestelmä***: Windows Vista Home Premium 64-bit; ***Python-versiot***: 2.4, 2.5 ja 2.6.

Lisäksi pelin toimivuutta on testattu SSH-yhteydellä tietojenkäsittelytieteiden laitoksen melkki-palvelimella (python 2.5) ja tietojenkäsittelytieteiden laitoksen tietokoneluokassa (python 2.6).

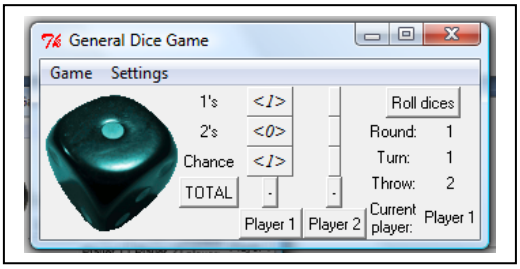
Peli-idea

Kyseessä on noppapeli, jossa tarkoituksena on noppia heittämälle kerätä mahdollisimman ison kokonaispistemäärä. Jos pelaajia on useampia tarkoituksena on saada enemmän pisteitä kun muut kilpailijat. Yksinpelissä tarkoituksena on saada mahdollisimman paljon pisteitä ja päästä mahdollisimman korkealle kaikkien aikojen top-10 -listalla. Noppia voi heittää yhdellä heittovuorolla niin monta kerta kun pelin asetuksissa on määritetty (1-10). Ensimmäistä heittokertaa lukuun ottamatta pelaaja voi heittää vain osan nopista. Kaikkia heittovuoroja ei tarvitse käyttää. Kun pelaaja on saanut mieleisensä yhdistelmän tai heittovuorot ovat loppuneet, pelaaja merkitsee heittonsa jonkun yhdistelmän kohdalle ja saa kokonaispistesaldoonsa noppien silmälukujen oikeuttaman pistesaldon. Pelin voittaja on se, jolla on pelin lopussa suurin kokonaispistesaldo. Jos voittajan pistesaldo on suurempi kuin pelin asetuksia vastaavan top-10 -listan huonoin pistesaldo, voittajan pistesaldo lisätään top-10 -listalle.

Ohjelman toiminnot

Pelissä esitellään seuraavat toiminnot:

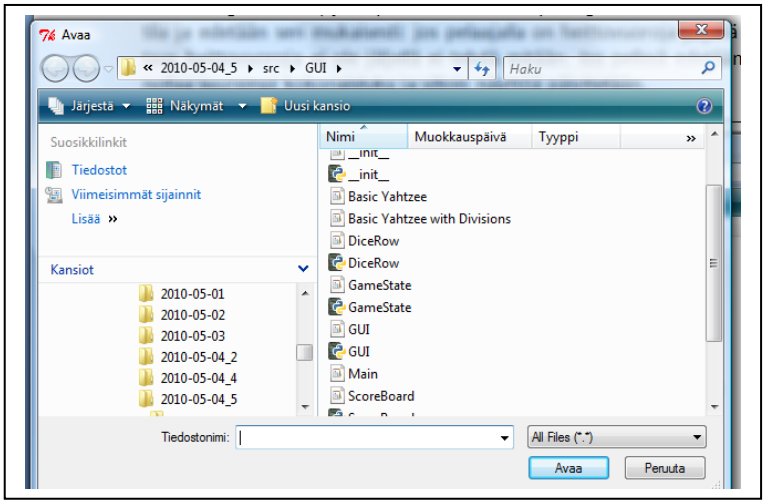
Nopan heittäminen ja uuden pelitiedoston avaaminen.



Kuva 1. Nopan heittäminen. Kuvan noppa on lukittu.

Nopan heittäminen

Noppaa heitetään painamalla napista 'Roll dices', valitsemalla valikosta 'Game' kohta 'Roll dices' tai painamalla näppäintä r (Kuva 1). Nämä kaikki toiminnot kutsuvat GameState-luokan root.states-olion funktiota goForward, joka puolestaan kutsuu pelin goForward funktiota. Tässä funktiossa tarkastetaan pelin tila ja edetään sen mukaisesti: jos pelaajalla on heittovuoroja jäljellä heitetään lukitsemattomia noppia, jos taas heittovuoroja ei ole jäljellä ei tehdä mitään. Jos pelissä edetään palautuu pelin goForward-funktiosta nollaa suurempi kokonaisluku ja silloin näyttöä päivitetään.



Kuva 2. Uuden pelitiedoston avaaminen (Windows Vista).

Uuden pelitiedoston avaaminen

Uusi pelitiedosto avataan valitsemalla 'Settings' valikosta kohta 'New Settings' (Kuva 1). Avautuvan valikon ulkonäkö riippuu käyttöjärjestelmästä (Kuva 2). Valikko tuotetaan tkFileDialog.askopenfilename-funktiolla, joka palauttaa tiedoston nimen. Pelin asetusten (game.settings) new_settings -funktiota kutsutaan tällä tiedostonnimellä. Tästä tiedostosta yritetään lukea yhdistelmiä. Jos yhtään yhdistelmää ei voida lukea tästä tiedostosta avataan oletustiedosto. Tämän jälkeen graafisen käyttöliittymän puolella tarkistetaan tuliko virheitä, ja jos tuli ne esitetään pelaajalle checkForErrors-funktion avulla tkMessagebox.showerror-funktiolla. Mahdolliset virheet on esitetty kohdassa virheilmoitukset (Taulukko 3, kolme ensimmäistä virheilmoitusta).

Syöttötiedot

Suurin osa käyttäjän antamasta syöttötiedosta on "idioottivarmaa" eli käyttäjä ei voi kaataa peliä tai antaa (pelin toiminnan kannalta) virheellistä syötettä. Pelissä edetään nappuloita painamalla. Pelaajan heittotavat

on esitelty aiemmin kohdassa 'Ohjelman toiminnot' - 'Nopan heittäminen'. Pelaaja valitsee yhdistelmän joko painamalla omaan tulostaulukon tulostaan hiirellä tai siirtymällä "TAB"-näppäimellä omien käyttämättömien tulostaulukon tulosten välillä. Pelaaja voi käyttää myös seuraavia pikanäppäimiä pelissä etenemiseen: 1,..,n (lukitse noppa 1,..,n); r (heitä); Escape, control-x, control-q (lopeta); control-n (uusi peli); control-p (uudet pelaajat) ja control-h (Top-10 -lista).

Tulostetiedot

Ohjelmaan kuuluu luokka Logger, jossa luodaan pelin lokitiedosto. Pelin lokitiedostoon kirjoitetaan joitakin pelin tärkeimpiä tapahtumia. Lisäksi on olemassa toinen lokitiedosto, joihin kirjoitetaan ainostaan peliin liittyvät virheilmoitukset ja varoitukset. Nämä kummatkin ovat kansiossa 'log'.

Lisäksi kansiossa 'Scores' sijaitsevat pelin top-10 -listat. Top-10 -listalle kirjoitetaan joka kerta kun peli päättyy. Pelin loputtua avataan kyseisen pelin top-10 -lista, jos sellainen on ja luetaan sen sisältö. Tämän jälkeen pelaajan pistesaldo lisätään listalle. Jos pelaajan tulos ei mahdu kymmenen parhaan joukkoon, poistetaan se listan lopusta. Lista tallennetaan pythonin pickle-moduulin avulla suoraan kirjastona (dict) ja luetaan kirjastona tiedostosta.

Taulukko 2. Peliin asetettuja rajoituksia noppien lukumäärälle (n), noppien maksimisilmäluvulle (s) ja heittojen lukumäärälle (h) sekä näiden parametrien oletusarvot.

Parametrin nimi	Arvo	Tulkinta
NMIN	1	Pienin mahdollinen määrä noppia, joita pelissä voi esiintyä.
NMAX	9	Suurin mahdollinen määrä noppia, joita pelissä voi esiintyä.
DEFAULT_N	5	Jos pelin asetustiedostossa ei anneta noppien lukumäärää, tai annettu lukumäärä n ei ole välillä NMIN≤n≤NMAX, käytetään tätä noppien lukumäärää
SMIN	2	Pienin mahdollinen noppien maksimisilmäluku, joka pelissä voi esiintyä.
SMAX	9	Suurin mahdollinen noppien maksimisilmäluku, joka pelissä voi esiintyä.
DEFAULT_S	6	Jos pelin asetustiedostossa ei anneta noppien maksimisilmälukua, tai annettu lukumäärä s ei ole välillä SMIN≤s≤SMAX, käytetään tätä noppien maksimisilmälukua
HMIN	1	Pienin mahdollinen heittojen lukumäärä kierrosta kohti, joka pelissä voi esiintyä.
HMAX	10	Suurin mahdollinen heittojen lukumäärä kierrosta kohti, joka pelissä voi esiintyä.
DEFAULT_H	3	Jos pelin asetustiedostossa ei anneta heittojen lukumäärää kierrosta kohti, tai annettu lukumäärä h ei ole välillä HMIN≤h≤HMAX, käytetään tätä heittojen lukumäärää kierrosta kohti

Ohjelman rajoitukset

Pelille on annettu asetustiedostossa muutama parametri, jotka rajoittavat pelin toimintaa (Taulukko 2).

Tämän lisäksi pelille voidaan antaa parametrinä suurin mahdollinen yhdistelmien lukumäärä, joka on sallittu. Graafisen käyttöliittymän käynnistyksen yhteydessä luotavalla pelillä tämä arvo on 30 eli pelissä on korkeintaan 30 yhdistelmää. Tämä tarkoittaa, että loput yhdistelmistä jätetään suoraan pois.

Graafinen käyttöliittymä lisää rajoituksia myös pelaajien nimiin: pelaajien nimet eivät voi olla tyhjiä tai sisältää ainoastaan välimerkkejä (white space) ja niiden maksimipituus on 21 merkkiä.

Taulukko 3. Tärkeimmät pelissä esiintyvät virheilmoitukset. Virheilmoitukset kirjoitetaan lokitiedostoon, tulostetaan päätteelle ja esitetään loppukäyttäjälle ponnahdusikkunoina.

Virheilmoitus	Selitys	Toiminta
`No dice combinations found from file '/home/pate/workspace/Yleinen noppapeli/src/GUI/DiceRow.pyc'! Loading default settings 'Basic Yahtzee.dat'.`	Pelille annetussa asetustiedostossa ei ole yhtään yhdistelmää.	Peli lataa automaattisesti oletusasetustiedoston. Jos haluat käyttää valitsemaasi tiedostoa, valitse uusi asetus-tiedosto tai lisää valitsemaasi asetustiedostoon yhdistelmiä.
`Given parameter for the number of dice (7e) is not a valid integer. Default value of 5 is used instead.`	Pelin asetustiedostossa oleva noppien lukumäärää olevaa parametria #n# seuraava merkkijono (7e) ei ole kään-nettävissä kokonaisluvuksi.	Korjaa asetustiedostossa olevan parametrin lukuarvoa. Tässä tapauksessa riittää poistaa e luvun 7 perästä.
`In the setting file '/home/pate/workspace/Yleinen noppapeli/src/GUI/Settings2.dat' there were no combinations that could be used with current settings. Loading default settings 'Basic Yahtzee.dat'.`	Pelille annetussa asetustiedostossa oli yhdistelmiä, mutta mitään niistä ei voitu ottaa käyttöön nykyisillä noppien luku-määrällä, maksimisilmä-luvulla ja heittojen luku-määrällä.	Peli lataa automaattisesti oletusasetustiedoston. Jos haluat käyttää valitsemaasi tiedostoa, poista tai muuta yhdistelmien perässä olevia rajoitteita.
`No code Found for code '123123'! Zero is returned.`	Valitun yhdistelmän koodia vastaavaa laskusääntöä ei löytynyt.	Tälle yhdistelmälle ei voi saada pisteitä ennen kuin koodia `123123` vastaava yhdistelmän laskusääntö lisätään Combinations-luokkaan.
`You did not select any players!`	Uusia pelaajia luotaessa käyttäjä ei ole valinnut (raksimalla) yhtään pelaajaa peliin tai kaikkien peliin valittujen pelaajien nimi on tyhjä.	Uusia pelaajia luotaessa, valitse pelaajille ei-tyhjä korkeintaan 21-merkkinen nimi (GUI:n rajoitus). Valitse vähintään yksi pelaaja raksimalla pelaajan nimen vieressä oleva laatikko.

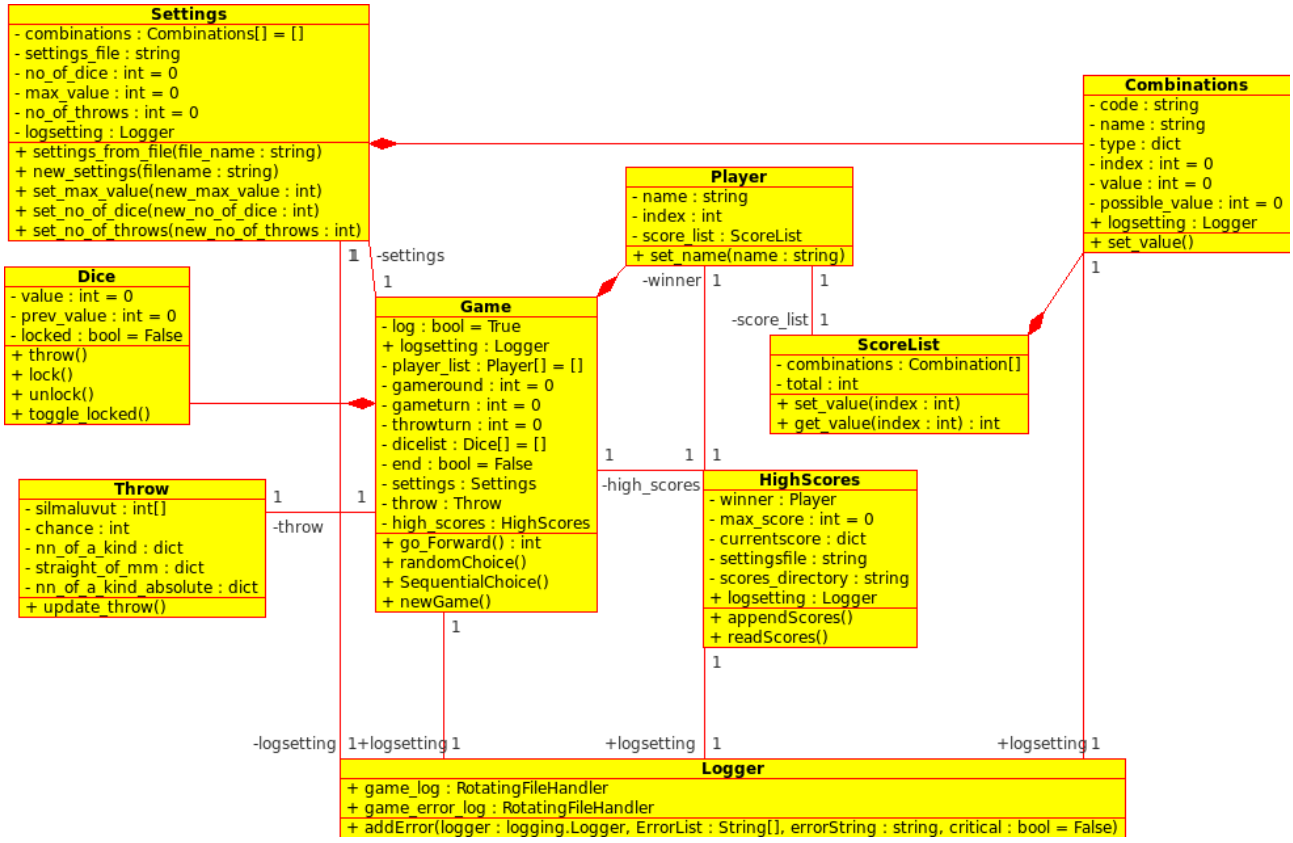
Virheilmoitukset

Suurin osa pelissä tapahtuvista virheistä korjataan valitsemalla huonolle parametrille parempi tai oletusarvo tai hylkäämällä huonot arvot. Pelin asetuksia voi muuttaa vain ennen ensimmäistä pelikierrosta. Näistä ”virheistä” toivutaan ja niihin ei kiinnitetä huomiota muuta kuin kirjataan pelin lokitiedostoon. Pelissä voi kuitenkin tapahtua muutama vakavampi virhe, joista on syytä ilmoittaa pelaajalle. Nämä virheilmoitukset kerätään peli-luokan muuttujaan `unNotifiedErrors`, josta ne siirretään ilmoituksen jälkeen muuttujaan `unNotifiedErrors`. Ohessa on näiden virheiden englanninkieliset virheilmoitukset, virheilmoitusten selitykset vaadittu toiminta (Taulukko 3).

Ohjelman toiminta ja rakenne

Yleiskuvaus

Luokkarakenne



Kuva 3. UML-kaavio yleisen noppapelin peruskomponenttien keskenäisistä suhteista. Luokkien attribuutit ovat merkitty yksityisiksi (-). Yksityisiksi merkittyihin attribuutteihin pääsee luokan ulkopuolelta käsiksi laittamalla alaviiva attribuutin nimen eteen. Kaikilla yksityisillä attribuuteilla on myös getterit (get_<attribuutin_nimi>), jotka on toteutettu pythonin property-ominaisuuden avulla. Vain "julkiseen" käyttöön eli toisten luokkien käyttöön tarkoitetut funktiot on esitetty. Sisäiseen käyttöön tarkoitettuja funktioita ei ole esitetty. Käännökset alkuperäisestä suomenkielisestä suunnitelmasta on esitetty liitteessä 1.

Pelin rakenteesta tuli melkolailla alkuperäisen suunnitteludokumentin suunnitelman mukainen, jos kielen vaihtoa ei oteta huomioon (Kuva 3). Uusia luokkia tuohon suunnitelmaan verrattuna ovat luokat HighScores (Top-10 -lista) ja Logger (loki). Esitetyssä luokkakaaviossa on vain luokkien julkiseen, eli toisten Pelin luokkien käyttöön tarkoitetut funktiot. Seuraavaksi luokittain lyhyt kuvaus luokkien välisistä suhteista ja luokkien tarkoituksista.

Game (Peli)

Peli-luokan funktiot on tarkoitettu pelin ulkopuolisen käyttöliittymän käyttöön. Kaikki luokat ovat riippuvaisia Peli-luokasta, eli sen kautta pääsee käsiksi kaikkiin pelissä oleviin olioihin siltä osin kun niihin on tarkoitus päästä käsiksi. Peli-luokan kautta edetään pelissä.

Dice (Noppa)

Noppa on yksinkertainen luokka, mutta erittän tärkeä, jossa on tieto nopan silmäluvusta ja siitä, onko noppa lukittu. Vain lukitsematonta noppaa voi heittää. Nopan silmälukua (value) voi vaihtaa vain heittämällä noppaa, eli arpomalla nopalle uuden silmäluvun yhdestä nopan maksimisilmälukuun.

Throw (Heitto)

Heitto-luokka on ikään kuin apuluokka Noppa-luokalle. Siinä lasketaan noppien kaikkien silmälukujen yhteisominaisuuksia, kuten silmälukulista, sattuma (chance), nn samaa (nn_of_a_kind), tasan nn samaa (nn_of_a_kind_absolute) ja mm:n suora (straight_of_mm). Näitä voidaan sitten käyttää hyväksi laskiessa eri yhdistelmien pisteitä. Heitto-luokan olio luodaan joka kerta uudelleen kun noppaa heitetään.

Settings (Asetukset)

Asetusluokan tehtävänä on käytännössä esittää pelin säännöt. Asetusluokan tärkeimmät attribuutit ovat noppien lukumäärä (no_of_dice, n), noppien maksimisilmäluku (max_value, s) ja heittojen maksimi-lukumäärä kierrosta kohti (no_of_throws, h). Nämä voidaan asettaa tämän luokan kautta. Kun niitä muutetaan tai asetukset luetaan ensimmäistä kertaa, luetaan pelin asetustiedostosta peliin edellä mainittujen attribuuttien sallimat yhdistelmät.

Combination (Yhdistelmä)

Yhdistelmä on käytännössä sääntö, jonka mukaan yhden heiton noppien silmäluvuista saa jonkun määrän pisteitä. Nämä säännöt on määritetty jokaiselle eri yhdistelmäkoodille erikseen tässä luokassa eikä uusia yhdistelmiä voi lisätä peliin muuttamatta pelin lähdekoodia. Yhtä yhdistelmäkoodia vastaan voidaan luoda a) yksi yhdistelmä ('SINGLE'), joka voi olla joko ehdoitta pelissä ('ALL'), vain peleissä, joissa on parillinen määrä noppia ('EVEN') tai pariton määrä noppia ('ODD') tai b) useampi yhdistelmä ('ENUMERATE'), joissa yhdistelmien lukumäärä riippuu joko noppien lukumäärästä ('NO_OF_DICE'), maksimisilmäluvusta ('MAX_VALUE') tai kummastakin ('MAX_VALUE_AND_NO_OF_DICE '). Näitä kolmea viimeeksi mainittua parametria kutsutaan jatkossa luetteloparametreiksi. Jos mitään muuta parametria ei ole annettu yhdistelmälle luetaan yhdistelmiä peliin, siten että ensimmäiselle yhdistelmälle annetaan indeksiksi 1 ja seuraavalle sitä suurempi lukuarvo kunnes saavutetaan annetun luettoloparametrin minimiarvo. Lisäksi tämän yhdistelmien indeksoinneille voidaan antaa minimi/maksimiarvot, joita ei voi alittaa/ylittää.

Yhdistelmälle voidaan antaa rajoitteita siitä, milloin yhdistelmä on käytössä. Nämä rajoitukset ovat suurimpia mahdollisia arvoja asetusluokan kuvauksessa mainituille kolmelle parametrille. Nämä rajoitukset voidaan asettaa kaikille yhdistelmille ja tälläistä rajoitusta rikotaan, ei yhdistelmäkoodi esiinny pelissä lainkaan. Esimerkki asetustiedostosta, jota on käytetty testauksessa on liittessä 2, indeksointiin liittyvien kommenttejen kera.

Player (Pelaaja)

Pelaaja on toistaiseksi hyvin yksinkertainen luokka, joka voitaisiin lähestulkoon jättään käyttämättä. Pelaajan luomisen yhteydessä tosin luodaan pelaajalla oma tulostaulukko. Pelaajalla on nimi ja järjestys-vuoro pelissä (index).

ScoreList (Tulostaulukko)

Tulostaulukkoon kirjataan pelaajan pisteet. Jokainen tulostaulukon tulosarvo liittyy saman indeksin omaavaan yhdistelmään pelin asetuksissa. Pelaajan omassa yhdistelmälistassa pidetään kirjaa siitä, mitkä yhdistelmät ovat jo käytettyjä ja mikä on pelaajan kokonaissaldo (total).

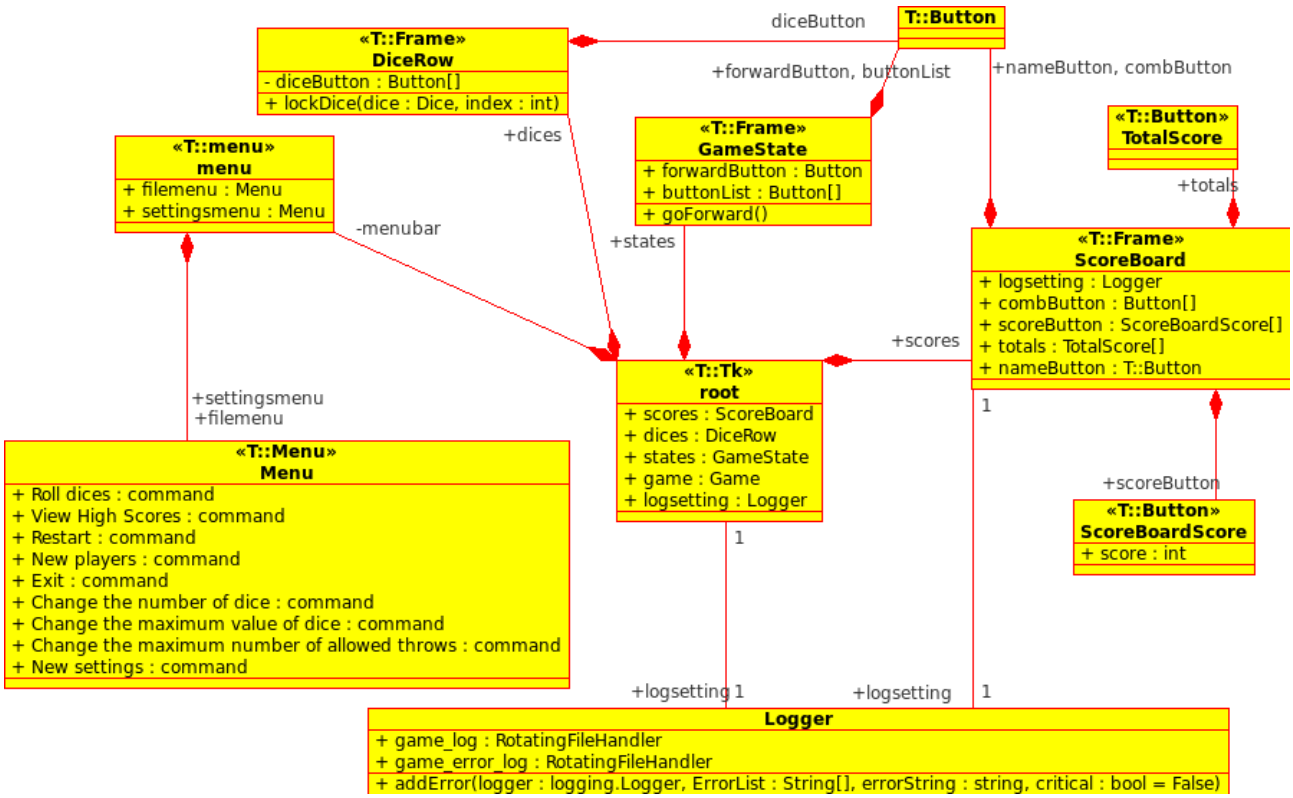
HighScores (Top-10 -lista)

Top-10 -listaa käytetään vain pelin lopussa kun tarkistetaan pääseekö pelin voittaja kaikkien aikojen top-10 -listalle.

Logger (Loki)

Lokin avulla kirjataan pelin tärkeimpiä tapahtumia tiedostoon virheenkorjausta varten. Lisäksi lokiin tallennetaan tieto pelissä tapahtuvista mahdollisista virhetilanteista (warning), sekä varsinaisista virheistä (error, critical). Varsinaiset virheet lisätään myös pelin käsittelemättömien virheiden listaan (unNotifiedErrors), josta käyttöliittymä voi tuoda niitä esiin (varoittaa käyttäjää) ja siirtää sitä mukaan käsiteltyjen listalle (NotifiedErrors).

Graafinen käyttöliittymä



Kuva 4. Luokkakaavio graafisen käyttöliittymän luokkien suhteista. T viittaa Tkinter-moduliin.

Graafisen käyttöliittymän toteutus tehtiin pitkälti suunnitteludokumentissa esitetyllä tavalla. Kaikkien graafisten käyttöliittymän kohteiden pohjana käytettiin valmiita Tkinter moduleita (Kuva 4). Graafisen käyttöliittymän pohjana on Tkinter.Tk -ikkuna nimeltä root, jonka päällä on kolme Tkinter.Frame-oliota, jotka taas muodostuvat Tkinter.Button-olioista. Lisäksi root-olioon liittyy valikko (Tkinter.menu), joka on näytön ylälaidassa. Valikon uudet ikkunat on toteutettu Tkinter.Toplevel-olioina, joiden sisällä on muita Tkinter-olioita kuten Tkinter.Button, Tkinter.Entry, Tkinter.CheckButton, Tkinter.Spinbox, Tkinter.Label ja Tkinter.LabelFrame. Lisäksi asetustiedoston avaamiseen käytetään tkFileDialog.askopenfilename-funktiota

ja virheistä tiedottamiseen ja toiminnan vahvistamiseen tkMessagebox.showerror ja tkMessageBox.askokcancel -moduuleita.

Graafisen käyttöliittymän toiminnot on hajoitettu pieniksi apufunktioiksi, jotka on kasattu moduliin GUI.

Luokkien kuvaus

Luokkien ja luokkien funktioiden yksityiskohtaisempi kuvaus löytyy python pyDocista (englanniksi).

Ohjeet ohjelman rajoitusten poistamiseksi

Pelissä ei oteta huomioon ikkunan mahtumista näyttön. Tästä syystä näytölle on annettu suosituskoko.

Parannusehdotuksia

Pelissä voitaisiin ottaa huomioon näytön koko. Yhdistelmien pisteytystä olisi myös hyvä voida säätää siten, että pisteytys voitaisiin lukea suoraan asetustiedostosta. Asetustiedostossa olisi myös hyvä voida käyttää vapaasti pelin vapaita parametreja n, s ja h (noppien lukumäärä, noppien maksimisilmäluku ja heittojen maksimilukumäärä kierrosta kohti). Lisäksi perinteisessä Yatzissa esiintyy bonus, jonka voi saada kun yläosan pisteet ovat tarpeeksi suuria. Ei ole kuitenkaan kovin selvää, miten tätä voisi yleistää vapaisiin yhdistelmiin. Lisäksi joissakin Yatzin 'virallisissa' säännöissä mainitaan bonukset joita saa, jos heittää useamman kuin yhden Yatzin.

Lisäksi näytön ulkoasuan voisi vielä hiukan kaunistella. Noppien värin voisi jättää käyttäjän päätettäväksi. Lisäksi olisi hyvä, jos jokainen pelaaja voisi tallentaa omat asetuksensa, jolloin oletuspelinä voitaisiin ladata edellinen peli ja pelaajille voitaisiin kerätä omaa top-10 -listaa.

Testauksen kuvaus

Testaus toteutettiin käyttämällä pythonin unittest-moduulia. Testauksessa toteutettiin kaksi eri testipakettia. Testaus voidaan suorittaa ajamalla paketissa Game oleva testiohjelma Unit_test_**2**.py. Testissä menee joitain kymmeniä sekunteja ja tulos tallennetaan tiedostoon `unit_test_2.txt`.

TestNoSettingsFunctions (Vapaiden parametrien testaus)

Testauksessa haluttiin ensin testata voiko pelin vapaille parametreille n, s ja h (noppien lukumäärä, noppien maksimisilmäluku ja heittojen maksimilukumäärä kierrosta kohti) antaa vain oikeita arvoja. Tätä testattiin valitsemalla joukko arvoja, jotka käsittävät kaikki sallitut arvot sekä vääriä arvoja (`[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 'string', {'dict': 'value'}]`). Testauksen aluksi luotiin peli. Tulosteessa esitettiin arvoa, joka yritettiin vapaalle parametrille syöttää sekä syöttämisen tulos, eli mikä oli pelin parametrin arvon syötteen antamisen jälkeen. Jokainen parametri käytiin läpi erillisenä testi-tapauksena. Testissä varmistettiin vielä, että lopullinen arvo on sallitulla välillä unittestin `self.assertTrue(Settings.SMIN <= self.game.settings.max_value <= Settings.SMAX)`-komennolla. Mahdolliset virheet esitettiin kaikkien vapaiden parametrien käsittelyn jälkeen. Saaduista tuloksista (Liite 4) nähdään, että parametrit saavat oikean arvon: parametrin arvo muuttuu jos parametri

on sallitulla välillä ja pysyy muuttumattomana, jos parametri on epäkelpo. Testiohjelma on toteutusdokumentin lopussa (Liite 3).

TestGameCompletion (Pelin läpiajamistesti)

Tässä testissä testataan, saako pelin pelattua läpi kaikilla mahdollisilla parametrin arvoilla. Testauksessa käytetään saama arvojoukkoa parametreille kuin vapaiden parametrien testauksessakin. Testissä kokeillaan kaikkia mahdollisia eri vapaiden parametrien yhdistelmiä. Testin alussa luodaan uusi peli, muutetaan sen vapaita parametreja ja mennään yksi askel eteenpäin (aloitetaan peli). Tämän jälkeen kokeillaan unittestin `self.assertFalse(self.game.end, "Game end was reached before the game started")`-komennolla, että peli ei varmasti ole päättynyt. Tämän jälkeen käydään läpi kaikki heittovuorot käymällä sisäkkäisissä loopeissa läpi kaikki pelaajat ja yhdistelmät. Tässä vaiheessa pelin pitäisi olla loppunut. Tämä tarkistetaan unittestin `self.assertTrue(self.game.end, "Game end was not reached")`-komennolla. Testin lopussa esitetetään vielä virheiden kokonaismäärä (0) ja testiin kulunut aika (60..70s). Lisäksi kun on ensin poistettu 'Scores'-kansiosta kaikki kansiot, nähdään, että peli kykenee luomaan peliasetuksille Scores-kansioon oman kansion ja luomaan sinne kaikille eri yhdistelmille top-10 -listat. Listojen kokonaismäärä oli testissä 720 kpl, joka on täsmälleen yhtä suuri kun kaikkien vapaiden parametrien arvoalueiden tulo, eli listoja luotiin oikea määrä ja kaikille parametriyhdistelmille. Lisäksi testitulosteesta (Liite 4) voidaan tarkastaa, että yhdistelmien lukumäärä vaihtelee parametrien (n ja s) mukaisesti.

Liitteet

Liite 1: Käännöksiä

Taulukko 4. Yleisimpiä ja tärkeimpiä pelissä käytettyjä termejä ja niiden käännökset suomesta englanniksi ja päinvastoin.

Suomeksi	Englanniksi
Arvo	Value
Asetukset	Settings
Heitto	Throw
Heittovuoro	Throw Turn
Graafinen käyttöliittymä	Graphical User Interface (GUI)
Indeksi	Index
Kokonaissaldo	Total
Koodi	Code
Loki	Logger
Lukitsematon	Unlocked
Lukossa	Locked
mm:n suora	Straight of mm
nn samaa	nn of a Kind
Nimi	Name
Noppa	Dice
Nopparivi	Dice Row
Pelaaja	Player
Peli	Game
Pelikierros	Game Round
Pelivuoro	Game Turn
Pelin tila	Game State
Sattuma	Chance
Silmäluku	Value
Top-10 -lista	High Scores
Tulostaulu (pelin)	Score Board
Tulostaulukko (pelaajan)	Score List
Tyyppi	Type
Yhdistelmä	Combination

Englanniksi	Suomeksi
Chance	Sattuma
Code	Koodi
Combination	Yhdistelmä
Dice	Noppa
Dice Row	Nopparivi
Game	Peli
Game Round	Pelikierros
Game State	Pelin tila
Game Turn	Pelivuoro
Graphical User Interface (GUI)	Graafinen käyttöliittymä
High Scores	Top-10 -lista
Index	Indeksi
Locked	Lukossa
Logger	Loki
Name	Nimi
nn of a Kind	nn samaa
Player	Pelaaja
Score Board	Tulostaulu (pelin)
Score List	Tulostaulukko (pelaajan)
Settings	Asetukset
Straight of mm	mm:n suora
Throw	Heitto
Throw Turn	Heittovuoro
Total	Kokonaissaldo
Type	Tyyppi
Unlocked	Lukitsematon
Value	Arvo, Silmäluku

Liite 2: Esimerkki asetustiedostosta

```
--------------------------------------------------------------------------

The Basic Yahtzee Game

Rules by Patrik Ahvenainen

Last update May 3, 2010

--------------------------------------------------------------------------

#N#        5          'Number of dice'
#s#        6          'Maximum value of dice'
#H#        3          'Maximum number of allowed throws'

#y#        000100     ''s'
#y#        000200     'straight of' nMin 4  sMin 4 indexMin 4     indexMax 5 [1]
#y#        000300     'of a kind'   nMin 5         indexMin 2     indexMax 4 [2]
#y#        000400     'two pairs'                                 nMin 4 [3]
#y#        000700     'Yahtzee'     nMin 5  sMin 3 [4]
#y#        001000     'Full House'  nMin 5 [5]
#y#        000000     'Chance'
```

———

Alla olevat kommentit eivät kuulu varsinaiseen tiedostoon (eivätkä yläindeksiviitteet)

[1] Vain suorat neljästä viiteen ovat olemassa ja vain jos noppia on vähintään neljä ja noppien maksimisilmäluku on vähintään neljä.

[2] n samaa ovat olemassa kahdesta (pari) neljään samaan, kunhan noppia on vähinttään viisi. Indeksi on rajoitettu neljään, jotta tämä yhdistelmä ei mene Yatzin (Yahtzee) kanssa päällekkäin.

[3] Erityisyhdistelmä, jota ei voi saada jos on alle neljä noppaa. Siksi neljän rajoitus indeksille.

[4] Yatzi (Yahtzee) saadaan perinteisesti kun on viisi noppaa heittämällä kaikilla nopilla sama silmäluku. Tällä yhdistelmällä olisi ehkä mielekästä olla lisärajoituksena indeksille `nMax 5`, jolloin yli viidellä nopalla pelatessa ei saisi helposti Yatzia.

[5] Erityisyhdistelmä, jota ei voi saada jos on alle viisi noppaa. Siksi viiden rajoitus indeksille.

Liite 3: Testiajokoodi

```
'''
Created on 2 May 2010
```

```
Last modified on 4 May 2010

@author: Patrik Ahvenainen
'''

import unittest
import Game
import Settings
import logging
import time
import sys


VALUE_TEST_LIST = [-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 'string', {'dict':
'value'}]
#VALUE_TEST_LIST = [3, 4, 5, 6, 7, 8, 9, 10, 11, 1000, 'string', {'dict':
'value'}]

SETTINGS_FILE = "Basic Yahtzee.dat"
WRITETOFILE = "unit_test_2.txt"
PLAYERS = "Player 1" , "Player 2"#, "Player 3", "Player 4"

WRITESTRING = ""

#class WriteStringClass(object):
#    def __init__(self):
#        self.WRITESTRING = ""



class TestGameCompletion(unittest.TestCase):

    def setUp(self):
        try: self.index1 == 0
        except: self.index1 = -1
        try: self.index2 == 0
        except: self.index2 = 0
        try: self.index3 == 0
        except: self.index3 = 0
        self.index1 += 1
        if self.index1 == len(VALUE_TEST_LIST):
            self.index2 += 1
            self.index1 = 0
        if self.index2 == len(VALUE_TEST_LIST):
            self.index3 += 1
            self.index2 = 0

        self.game = Game.Game(PLAYERS, SETTINGS_FILE, log=False)
        self.game.settings.max_value = VALUE_TEST_LIST[self.index1]
        self.game.settings.no_of_dice = VALUE_TEST_LIST[self.index2]
        self.game.settings.no_of_throws = VALUE_TEST_LIST[self.index3]
        self.game.goForward()

    def test_complete_game(self):
        self.assertFalse(self.game.end, "Game end was reached before the game
started")
        for no_of_players in range(len(self.game.player_list)):
            for i in range(0, len(self.game.settings.combinations)):
                self.game.SequentialChoice()
        if self.game.end:
```

```
            print "Game settings are: n= " + str(VALUE_TEST_LIST[self.index1])
+ "-->" + str(self.game.settings.no_of_dice) + \
                  ", s=" + str(VALUE_TEST_LIST[self.index2])  + "-->" +
str(self.game.settings.max_value) + \
                  ", h=" + str(VALUE_TEST_LIST[self.index3])  + "-->" +
str(self.game.settings.no_of_throws) + \
                  ", no of combinations=" +
str(len(self.game.settings.combinations))
        self.assertTrue(self.game.end, "Game end was not reached")

    def tearDown(self):
        logging.shutdown()

class TestNoSettingsFunctions(unittest.TestCase):

    def setUp(self):
        self.game = Game.Game(PLAYERS, SETTINGS_FILE, log=False)

    def test_set_max_value(self):
        print "\nTESTING 'MAX VALUE OF DICE' ASSIGNMENT"
        print "ACCEPTABLE VALUES ARE BETWEEN " + str(Settings.SMIN) + \
              " AND " + str(Settings.SMAX)
        for testItem in VALUE_TEST_LIST:
            self.game.settings.max_value = testItem
            print "Trying to set value '" + str(testItem) + \
                  "'. New value is = " + str(self.game.settings.max_value)
            logging.shutdown()
        self.assertTrue(Settings.SMIN <= self.game.settings.max_value <=
Settings.SMAX)

    def test_set_no_of_dice(self):
        print "\nTESTING 'NUMBER OF DICE' ASSIGNMENT"
        print "ACCEPTABLE VALUES ARE BETWEEN " + str(Settings.NMIN) + \
              " AND " + str(Settings.NMAX)
        for testItem in VALUE_TEST_LIST:
            self.game.settings.no_of_dice = testItem
            print "Trying to set value '" + str(testItem) + \
                  "'. New value is = " + str(self.game.settings.no_of_dice)
            logging.shutdown()
        self.assertTrue(Settings.NMIN <= self.game.settings.no_of_dice <=
Settings.NMAX)

    def test_set_no_of_throws(self):
        print "\nTESTING 'MAXIMUM NUMBER OF THROWS' ASSIGNMENT"
        print "ACCEPTABLE VALUES ARE BETWEEN " + str(Settings.HMIN) + \
              " AND " + str(Settings.HMAX)
        for testItem in VALUE_TEST_LIST:
            self.game.settings.no_of_throws = testItem
            print "Trying to set value '" + str(testItem) + \
                  "'. New value is = " + str(self.game.settings.no_of_throws)
            logging.shutdown()
        self.assertTrue(Settings.HMIN <= self.game.settings.no_of_throws <=
Settings.HMAX)

    def tearDown(self):
        logging.shutdown()

if __name__ == '__main__':
#    unittest.main()'
    no_of_total_errors = 0
```

```
    sys.stdout = open(WRITETOFILE, 'w')
    print ("Starting unit tests\n")

    totaltime = time.clock()

    test_suite = unittest.TestSuite()
    test_set_max_value_ = TestNoSettingsFunctions('test_set_max_value')
    test_set_no_of_dice_ = TestNoSettingsFunctions('test_set_no_of_dice')
    test_set_no_of_throws_ = TestNoSettingsFunctions('test_set_no_of_throws')
    test_suite.addTest(test_set_max_value_)
    test_suite.addTest(test_set_no_of_dice_)
    test_suite.addTest(test_set_no_of_throws_)
    testresults = unittest.TestResult()
    timeri = time.clock()
    test_suite.run(testresults)
    took = time.clock() - timeri
    if len(testresults.errors) > 0:
        no_of_total_errors += 1
    print ( "\nErrors: " + str(testresults.errors))
    if testresults.wasSuccessful():
        print ( "SUCCESS in " + str(took) + "s !" )

#    myObj = WriteStringClass()

    print "\nTESTING 'GAME COMPLETION' ASSIGNMENT USING RANDOM SELECTION OF
SCORES TO MARK, "
    print "TEST SETTINGS FILE '" + str(SETTINGS_FILE)
    timeri = time.clock()
    test_suite = unittest.TestSuite()
    test_complete_game = TestGameCompletion('test_complete_game')
    testresults = unittest.TestResult()
    for index, item1 in enumerate(VALUE_TEST_LIST):
        for item2 in VALUE_TEST_LIST:
            for item3 in VALUE_TEST_LIST:
                test_suite = unittest.TestSuite()
                test_suite.addTest(test_complete_game)
                test_suite.run(testresults)
                if len(testresults.errors) > 0:
                    no_of_total_errors += 1
                if len(testresults.errors) > 0:
                    print  ", Errors: " + str(testresults.errors)
                if testresults.wasSuccessful():
                    print  ", SUCCESS"
#        print "Game completion test %4.2f completed in %3.1fs" %
((index+1)/float(len(VALUE_TEST_LIST)), time.clock() - totaltime)

    print "\nIn total the test took " + str(time.clock() - totaltime) + "s."
    print "There were " + str(no_of_total_errors) + " errors in total"

    sys.stdout = sys.__stdout__
```

Liite 4: Testiajon tulokset

Tiedostossa testiajon_tulokset suuren koon vuoksi.