

2022

Ohjelmoinnin peruskurssin harjoitustyö

DOKUMENTOINTI

HENRIK KOSKINEN 2206392

Sisällys

1. Työn kuvaus ja aihe.....	1
2. Rakenne ja funktiot.....	1
1. new_day(self):.....	1
2. power(self):.....	1
3. rested(self):.....	1
4. health(self):.....	1
5. configure_stats(self):	2
6. check_die(self,prompt):.....	2
7. add_loss(self):	2
8. check_values(self,value):	2
9. victory(self):	2
10. reset(self):	2
11. ja 12 eli stop(self): ja start(self):	2
13. rulest_reset(self):.....	2
14 game_rules(self):	3
3. Ulkoiset kirjastot	3
4. Lisätiedot.....	3

1. Työn kuvaus ja aihe

Työn aiheena on peli, missä tavoitteena on saada pelaajan Strength(voima) arvo 9000:en antamalla graafiselle käyttöliittymälle arvoja pelissä annetulla välillä. Arvot lasketaan aina "New day" nappulan painamisen jälkeen pelaajan sen hetkisiin arvoihin (Status). Pelin säännöt selitetään tarkemmin pelin sisällä painamalla "Game rules" nappulaa. Työn tarkistajat katsokaa kohdasta "Lisätiedot" dokumentit lopussa miten Statusten vähenemiset toimivat.

2. Rakenne ja funktiot

Työssä käytetään pitkälti class Player ja sen sisäisiä arvoja. Miksi classia käytetään? Ajatuksena oli, että jos tätä jatkaa ja pidentää sekä parantaa myöhemminkin kuin kurssilla, niin on parempi rakentaa se classin ympärille, sillä se on valmiiksi object orientoitua ja helpottaa isomman rakenteen tekemistä sekä muokkaamista. Classiin on rakennettu, arvojen lisäksi, myös koko GUI eli kaikki nappulat, taustat ja otsikot löytyvät sieltä.

Työssä on 14 funktiota:

1. new_day(self):

Funktio ei tarvitse syötteitä ja se käy läpi tarvittavat funktiot ja nollaa taulun käyttäjän kirjoittamista arvoista, jotta käyttäjä voisi antaa uudet arvot puhtaalle taululle.

2. power(self):

Funktio ottaa annetun arvon käyttäjän tauluun kirjoittamasta arvosta käyttämällä .get() komentoa. Sekä Pain! että Strength kohdan arvot otetaan ja tarkistetaan funktiossa check_values() mikäli ne ovat virheellisiä. Tämän jälkeen ne käydään läpi ja lisätään käyttäjän sen hetkisiin arvoihin tai vähennetään mikäli arvot olivat tyhjiä/virheellisiä ja tarkistetaan victory() funktiolla mikäli voitto on saavutettu.

3. rested(self):

Otetaan käyttäjän syöttämät arvot taululta kohdasta Rested ja käydään se läpi. Mikäli on tullut nukuttua huonosti eli alle 6 niin huonosti nukuttujen päivien määrää kasvatetaan yhdellä. Tämä on yksi häviö tekijöistä, eli jos niitä tulee 3 perätyksen niin peli loppuu.

4. health(self):

Alustetaan maksimiarvo (=15) ja tarkistetaan arvot kohdista Hydration ja Health sekä käydään en läpi. Mikäli ei anna kunnollista tai antaa tyhjän arvon kohtaan Hydration Entry, niin Status Hydrationia vähennetään viidellä (=- 5) New day nappulaa painattaessa. Samaa käytetään kohdassa Health missä arvoa vähennetään samoin perustein kymmenellä (=- 10).

5. `configure_stats(self)`:

Tämä funktio muuttaa tarvittaessa GUI:ssa näkyviä tekstejä sen mukaan mitä pelissä on tapahtunut. Jos kuolee koska Health tippuu nolleen, niin se muuttaa tekstin punaiseksi vihreän sijaan ja mikäli arvot menevät yli 100 tiputtaa se arvon takaisin 100 ennen kuin päivittää sen taululle.

Mikäli on nukkunut huonosti kolmesti putkeen niin Strength puolitetaan sekä riippuen muista rested arvoista, muokataan Status tekstiä joko keltaiseksi (kun `badly_slept` on 3 tai yli) ja vaihdetaan teksti Tired. Mikäli arvo `no_sleep` on kolme, niin muokataan Status Rested punaiseksi ja teksti Exhaustediksi.

Samaa periaatetta käyttäen Hydrationia muokataan, muuttaen sen väriä ja tekstiä, mikäli status tippuu alle 50 ja pahimmassa tapauksessa alle 0.

Myös Pain! Kohdan ollessa 7 pelaaja kuolee ja siksi se muutetaan punaiseksi, jotta käyttäjä näkee syyn häviölleen. Sama pätee myös kohdalle Strength, mutta tässä pelätään arvoa 0 sen sijaan.

6. `check_die(self,prompt)`:

Funktio tarkistaa mikäli jonkun häviön tunnusmerkit ovat täyttyneet ja antaa pienen ilmotuksen käyttäjälle, sekä poistaa New day nappulan käytöstä

7. `add_loss(self)`:

Tavallaan hyödytön, mutta mahdollista suurentaa tarvittaessa. Lisää häviöiden lukumäärään yhden ja configuroi sen taululle.

8. `check_values(self,value)`:

Tärkeimpiä funktioita, tarkistaa mikäli arvo on kokonaisluku ja yli nollan, muuten palauttaa arvon tyhjänä.

9. `victory(self)`:

Tarkistaa mikäli käyttäjä on saanut Strength Status arvon 9000:en ja muokkaa tekstiä sekä antaa pienen onnitteluviestin.

10. `reset(self)`:

Resetoi taulut tyhjäksi ja antaa käyttäjälle puhtaan pöydän, paitsi voitot ja häviöt eivät muokkaannu. Niistä ei pääse eroon, ellei sulje peliä.

11. ja 12 eli `stop(self)`: ja `start(self)`:

Stop lopettaa pelin ja start aloittaa pelin. Syy miksi laitoin nämä erikseen oli se, että mikäli haluan laittaa lisää toimintoja loppuun ja alkuun niin on jo valmiiksi funktiot mihin lisätoiminnot voi rakentaa.

13. `rulest_reset(self)`:

Tuhoaa tarvittaessa Game rules ikkunan ja estää käyttäjää avaamasta 1000000 game rules ikkunaa samanaikaisesti, mikä oli ongelmana tosi pitkään.

14 game_rules(self):

Luo uuden ikkunan, missä on pelin säännöt

3. Ulkoiset kirjastot

Ainut ulkoinen kirjasto mitä tässä työssä käytetään on tkinter, koska sitä tarvitaan Pythonissa GUI tekemiseen. Sieltä saa tarvittavat moduulit ja widgetit. Esim. Tk() toplevel ikkunan tekemiseen, button nappuloiden luomiseen.

4. Lisätiedot

Statusten väheneminen:

Mikäli käyttäjä ei anna arvoja niin Strenght vähenee 200:lla, Health 10:llä ja Hydrated 5:llä.

Ohjelmassa ei käytetä sanakirjaa(lista löytyy) tai lueta/kirjoiteta tiedostoihin, jotenka toivon saavani vapautuksen näistä vaatimuksista, sillä työ on jo itsessään hyvin pitkä(508 riviä) sekä hieman vaativampi ja en nähnyt niille tarvetta taikka käyttöä.