Tekninen suunnitelma

1. Ohjelman rakennesuunnitelma

Luokat:

Note: nuotti. Attribuutteja mm. korkeus, oktaavi, aloitushetki, kesto. Metodeja esim. get_pitch(), joka palauttaa nuotin korkeuden.

Rest: tauko. Attribuutteja kuten yllä.

LyricSyllable: sanoitustavu, samoja attribuutteja kuin aiemmilla, lisäksi tietysti tavu itse.

Composition: yläluokka, joka kokoaa olioon kaikki annetut sävelmän osat. Attribuutteja mm. kappaleen tiedot ja kaikki nuotti-, tauko- ja sanoitustavuoliot. Metodeja esim. add_note(), add_rest().

ChunkIO: lukee tekstitiedoston, lataa ja tallentaa annetut tiedot. Luo uuden Composition-olion, ja siihen kaikki annetut pienemmät oliot. Metodeja: load_data(input) lataa tiedostosta ja käyttäjän syötteestä tiedot käyttämällä apumetodeja parse_note(input), parse_rest(input) jne. jotka luovat tietojen perusteella nuotti- ym. –olioita.

CharGraphics: Tulostaa nuottiviivastolle annetut nuotit merkkigrafiikkana. Tärkein metodi on create_table, joka luo taulukon (Compostion-olion perusteella) joka on mahdollista tulostaa. Käyttää apuluokkaa:

Position: antaa nuotille, tauolle ja sanoitustavulle niiden attribuuttien mukaisen sijainnin graafisessa nuottirivistössä.

2. Käyttötapauskuvaus

Ensimmäisenä ohjelma kysyy käyttäjältä tiedostoa, jossa on vähintään sävelmän tiedot-lohko. Tiedostossa on tässä esimerkissä muutama nuotti ja taukoja. Ohjelma lukee tiedoston ChunkIO-luokassa ja tallentaa sävelmän tiedot Composition-olioksi. Tämä olio käyttää luokkia Note, LyricSyllable ja Rest.

Composition olio luo itselleen myös uuden CharGraphics-olion, joka tulostaa nuottiviivaston merkkigrafiikkana. Tämä luokka käyttää apuluokkaa Position.

Tulostuksen jälkeen käyttäjä voi luoda uusia nuotteja, taukoja tai sanoituksia sävellykseensä, tai poistaa niitä. Käyttäjä antaa samat tiedot suoraan ohjelmalle samalla syntaksilla kuin tekstitiedostossa, mutta rivi kerrallaan. Tietojen tallennus ja tulostus tapahtuu suurin piirtein samaan tapaan kuin tiedostoa luettaessa.

3. Algoritmit

Luokassa ChunkIO tarvitaan algoritmi, joka lukee tiedostoa. Metodi lukee tekstitiedostoa rivi kerrallaan ja etsii tiettyjä merkkijonoja.

Luokka CharGraphics tulostaa nuottiviivaston käyttämällä apuna esimerkiksi moduulia astropy.io.ascii [1], jonka avulla voi tulostaa merkkitaulukoita.

4. Tietorakenteet

Jokaisella nuotilla CharGraphics-oliossa on sijainnin määrittämiseen Position-olio, jolla on attribuutteina tahti, korkeus ja aloitushetki. Position-oliot ovat moniulotteisessa listassa.[2] On hyvä käyttää listaa, koska sävelmä voi olla mielivaltaisen pituinen ja sisältää eri määrän nuotteja, taukoja ja sanoituksia.

Position-oliot sisältävän listan tiedot muutetaan sitten astropy.io.ascii-moduulin käyttämän syntaksin mukaiseksi tulostusta varten.

5. Aikataulu

Aloitan laatimalla luokkarakennekehyksen; 3-5h. Tämän jälkeen varmaan huomaa mitkä luokat kannattaa toteuttaa ensimmäisenä. Aikaa eniten vaativin osuus vaikuttaisi olevan tulostuksen muotoilu, joten mietin todennäköisesti seuraavaksi jonkin verran sen toteutusta. Yhteensä siihen tulee kulumaan suurin osa ajasta, karkea arvio: 10-30h. Toinen tärkeä osa ohjelmaa on käyttäjän antamien tietojen parsiminen; 10h. Loppuaika jakautuu tasaisesti kaikkien osien kesken; toukokuuhun saakka n. 10h/viikko.

6. Yksikkötestaussuunnitelma

ChunkIO-luokan metodit parse_note(), parse_rest() jne. ovat keskeisimpiä testausta vaativia ohjelman osia. Näitä voidaan testata virheellisillä tiedostoilla ja käyttäjä-inputeilla. Esimerkiksi puuttuva tiedot-lohko, vääränlaiset nuottien korkeudet ja sijainnit tai vääränlaiset tiedot tulisi nostaa virheilmoitus eikä tulostaa merkkigrafiikkaa. Tätä voi testata antamalla ohjelmalle sellaisia tiedostoja.

Toinen tärkeä osa on CharGraphis-luokka, jossa Composition-olion tiedot täytyy muuttaa muotoon joka tulostaa oikeanlaisen nuottirivistön. Luokassa on metodi create_table() joka luo ascii-taulukon, jonka astropy.io.ascii moduulin metodit osaavat käsitellä ja tulostaa.

7. Kirjallisuusviitteet ja linkit

[1] astropy.io.ascii-moduuli:

http://hea-www.harvard.edu/~aldcroft/tmp/p4a/hamogu/html/files/asciifiles.html#astropy-io-ascii

http://docs.astropy.org/en/v0.2.1/io/ascii/

[2] List Comprehensions:

https://docs.python.org/3/tutorial/datastructures.html#list-comprehensions