Dokumentti: Nuottipiirturi Python-ohjelma
Y2 Ohjelmoinnin peruskurssi
Tekijä: Anssi Moisio 474694, EST/ELEC vuosikurssi 2
5.5.2017

2. Yleiskuvaus

Ohjelma, jolla voidaan kirjoittaa yksinkertaisia sävelmiä nuottiviivastolle ja esittää nuotit tekstimuodossa. Tällaiselle ohjelmalle on käyttöä, jos vaikkapa haluaisi liittää yksinkertaisen melodian nuotit sähköpostiin. Ohjelma lukee ja kirjoittaa tekstitiedostomuodossa tallennettuja sävelmiä.

Ohjelma täyttää vaikean tehtävänannon vaatimukset lukuunottamatta tahtirajan ylittäviä nuotteja.

3. Käyttöohje

Ohjelma pyytää ensimmäisenä käyttäjältä tekstitiedostoa, johon on tallennettu sävelmä. Tekstitiedosto tulee alkaa sanoilla ’Savellys X.X tallennustiedosto’. Tekstitiedostossa on lueteltu sävellyksen tiedot, nuotit, tauot, palkit ja sanoitus omissa lohkoissaan, esim:

```
#TIEDOT
tekija: Jaakko
Nimi: Opus no:54 Nocturne in B-flat
alennetut: B,A,F

#NUOTIT
G,0,e,e, 0, 1/8, 1/8
F,0, e, k, 0, 2/8, 1/8

#TAUOT
1, 1/8,1/4
3,1/8,1/4

#PALKIT
0, 0:1
1,0:1

#SANOITUS
0, 3/8, Hel
0, 4/8, lo
```

Tiedoston muoto ja syntaksi on selitetty tarkemmin luvussa 7, johon kannattaa tutustua ennen ohjelman käyttämistä.

Tärkeimmät ominaisuudet:

Ohjelma osaa piirtää nuotteja joiden kesto on 1, 1/2, 1/4, 1/8, 1/16 tai näistä puolikkaan keston verran pidennettyjä pisteellä, esim 3/2.

Ohjelma osaa näyttää alennetut, korotetut ja palautetut nuotit, merkeillä ’b’, ’#’ ja ’X’. Koko sävelmän ajan korotetut tai alennetut nuotit luetellaan

lohkoon #TIEDOT ja yksittäiset nuotit merkitään nuotteja lisätessä merkinnällä k/e (kyllä tai ei). Esim F,0, e, k, 0, 2/8, 1/8 on korotettu nuotti ja ylemmässä esimerkissä koko sävelmän ovat alennettuja ovat nuotit B, A ja F. Koko sävelmän ajan korotetut tai alennetut nuotit merkitään nuottiavaimen (engl. clef) jälkeen.

Ohjelma osaa piirtää taukoja joiden kesto on 4, 2, 1, 1/2, 1/4, 1/8 tai 1/16.

Ohjelma osaa piirtää palkkeja (engl. beam) 1/8- (tai 3/16-)kestoisille nuoteille. Palkeilla voi tehdä nuottiviivastosta helpommin luettavaa.

Ohjelma tulostaa tiedoston perusteella nuottiviivaston, jonka jälkeen sävellyksestä voi poistaa tai siihen lisätä nuotteja, taukoja tai palkkeja, tai muuttaa tietoja. Tämä tapahtuu näppäimistö-inputilla, syöttämällä ohjelmalle komentoja, esimerkiksi ’lisaa tauko’, jonka jälkeen ohjelma antaa syntaksin tauon määrittämiseen. Syntaksi on merkintöjen lisäämiseen sama kuin tallennustiedostossa ja se on selitetty tarkemmin luvussa 7.

Tyhjät kohdat sävellyksessä voi täyttää tauoilla komennolla ’tayta’.

Muutokset voi tallentaa antamalla komennon ’tallenna’, minkä jälkeen ohjelma pyytää tallennukseen käytettävää tiedostoa, joka voi olla tai olla olematta sama tiedosto kuin alkuperäinen luettava tiedosto.

Kaikki mahdolliset komennot:

lisaa nuotti
poista nuotti
lisaa tauko
poista tauko
lisaa palkki
poista palkki

Yllä olevat komennot jättävät ohjelman lisäyksen jälkeen tähän moodiin, jossa ohjelma pyytää aina seuraavaa lisättävää/poistettavaa nuottia/taukoa/palkkia. Tästä moodista voi poistua komennolla ’esc’, jolloin ohjelma tulostaa nuottiviivaston.

tayta
tallenna
quit

Komento ’quit’ lopettaa ohjelman suorituksen. Tallentamattomat tiedot katoavat.

Tässä esimerkki siitä, mitä komentoja antamalla ohjelma tekee järkeviä asioita. Kahden ensimmäisen rivin pitäisi käynnistää ohjelman suoritus:

```
cd Y2_Nuottipiirturi\scr
python main.py
yesterday.txt
muuta tietoja
pituus:9
esc
lisaa nuotti
A, 1, e, e, 7, 1/2, 1/2
A, 1, e, e, 7, 1/1, 1/2
esc
lisaa nuotti
D, 1, e, e, 8, 1/4, 1/4
E, 1, e, e, 8, 1/2, 1/4
F, 1, e, e, 8, 3/4, 1/4
E, 1, e, e, 8, 7/8, 1/8
D, 1, e, e, 8, 1/1, 1/8
C, 1, e, e, 8, 1/1, 1/8
esc
poista nuotti
8,5
esc
lisaa palkki
8, 3:4
esc
tallenna
uusi1.txt
quit
```

Tämän jälkeen ohjelman voi käynnistää uudelleen ja kokeilla antaa ohjelmalle tiedoston uusi1.txt

4. Ohjelman rakenne

Ohjelma jakautuu kahdeksaan eri luokkaan. Luokat Note, Rest, Beam ja Lyrics luovat nuotti-, tauko-, palkki-, ja sanoitustavuoliot, jotka tallennetaan ja kootaan listoihin luokassa Composition. Nuottiviivaston piirtäminen merkkigrafiikkana tapahtuu luokissa CharGraphics ja Column, joista jälkimmäinen kuvaa yhtä saraketta viivastossa, johon mahtuu esimerkiksi yksi tauko.

Käyttöliittymä ja tekstitiedon lukeminen ja parsiminen tapahtuvat luokassa TextFileIO.

Luokat:

Note: Nuotti. Attribuutteja: korkeus, oktaavi, alennus (Boolean), ylennys (Boolean), tahti, aloitushetki, kesto

Rest: Tauko. Attribuutteja: tahti, aloitushetki, kesto

Beam: Palkki. Yhdistää nuotteja, auttaa rytmittämään sävellystä. Attribuutteja: lista nuoteista jotka tämä palkki yhdistää, tahti, korkeus viivastolla, aloitushetki, lopetushetki. Muut attribuutit määritellään luokan sisällä nuottilistan avulla ja sen perusteella.

Lyric: sanoitustavu. Attribuutteja: tahti, aloitushetki, merkkijono

Column: Tulostettava esimerkiksi 1/8 –tahdin kokoinen sarake, 16*6 kokoinen kaksiulotteinen lista merkkejä. Riveillä 2,4,6,8,10 on viivat ’------’ ja näiden väliin white space. Lista saa täytettä nuoteista - ja näihin liittyvistä palkeista - tauoista ja sanoituksista (alin rivi 15). Esimerkiksi kahdeksasosanuotti A# oktaavissa 1, ja sen päällä ’stem’ ja ’flag’, alla yksi tavu sanoitusta:

```
------
   \
---|\-
   |/
---|--
# @@
------

------


 Moi
```

näitä asetetaan sitten peräkkäin, jolloin muodostuu koko tahti:

```
     /'\
 ----|--/---------------------=======----------------------------|
     | /                     |     |     |                        |
 ----\/----------------------|-----|-----|------------------------|
     /\          |           |     |     |                        |
 ---/-|------b---|-----------|-----|-----|------------------------|
   / _|_         |          @@    @@    OO                        |
 -|-\-|-\--------|-------------------------------------------------|
  \___|_/       @@                                                |
 ------\-----------------------------------------------------------|
   @@   \
    \___/


              yes -       ter - day._______
```

Metodi find_items() etsii Composition-oliosta kaikki tässä sarakkeessa olevat merkinnät. Muut tämän luokan metodit piirtävät merkinnät Column olion merkkimatriisiin (rows).

Myös tällä oliolla on attribuutteina tietysti tahti ja aloitushetki.

Composition: yläluokka, joka kokoaa olioon kaikki annetut sävelmän osat. Nuotit, tauot, palkit ja sanoitukset ovat omissa listoissaan. Sävellyksen nimi ja tekijä attribuutteina.

Metodi fill_holes() täyttää tyhjät kohdat sävellyksessä tauoilla, joiden kesto on sama kuin tahdissa oleva kestoltaan lyhyin nuotti tai tauko.

TextFileIO: lukee tekstitiedoston, lataa ja tallentaa annetut tiedot. Luo uuden Composition-olion, ja siihen kaikki annetut pienemmät oliot. Luo CharGraphics olion, joka piirtää nuottiviivaston.

Metodi read_file() lataa tiedostosta tiedot ja metodi keyboard_input() muokkaa tiedostoa käyttäjän syötteen mukaisesti käyttämällä apumetodeja parse_note(), parse_rest() jne., jotka luovat käyttäjän antamien tietojen perusteella nuotti- ym. -olioita.

Metodi write_file() pyytää käyttäjältä tekstitiedoston nimen ja tallentaa sävellyksen tähän tiedostoon samalla syntaksilla jota ohjelma pystyy myös lukea uudestaan.

Sort_measurenotes(measure) on apumetodi, joka järjestää yhden tahdin nuotit järjestykseen alkamisajan ja sävelkorkeuden perusteella. Jos nuotit ovat samalla hetkellä, korkeampi nuotti on ensin. Tämä on tarpeellista, jotta käyttäjä voi poistaa tietyn nuotin tämän järjestyluvun avulla.

CharGraphics: Tulostaa nuottiviivastolle annetut nuotit merkkigrafiikkana.

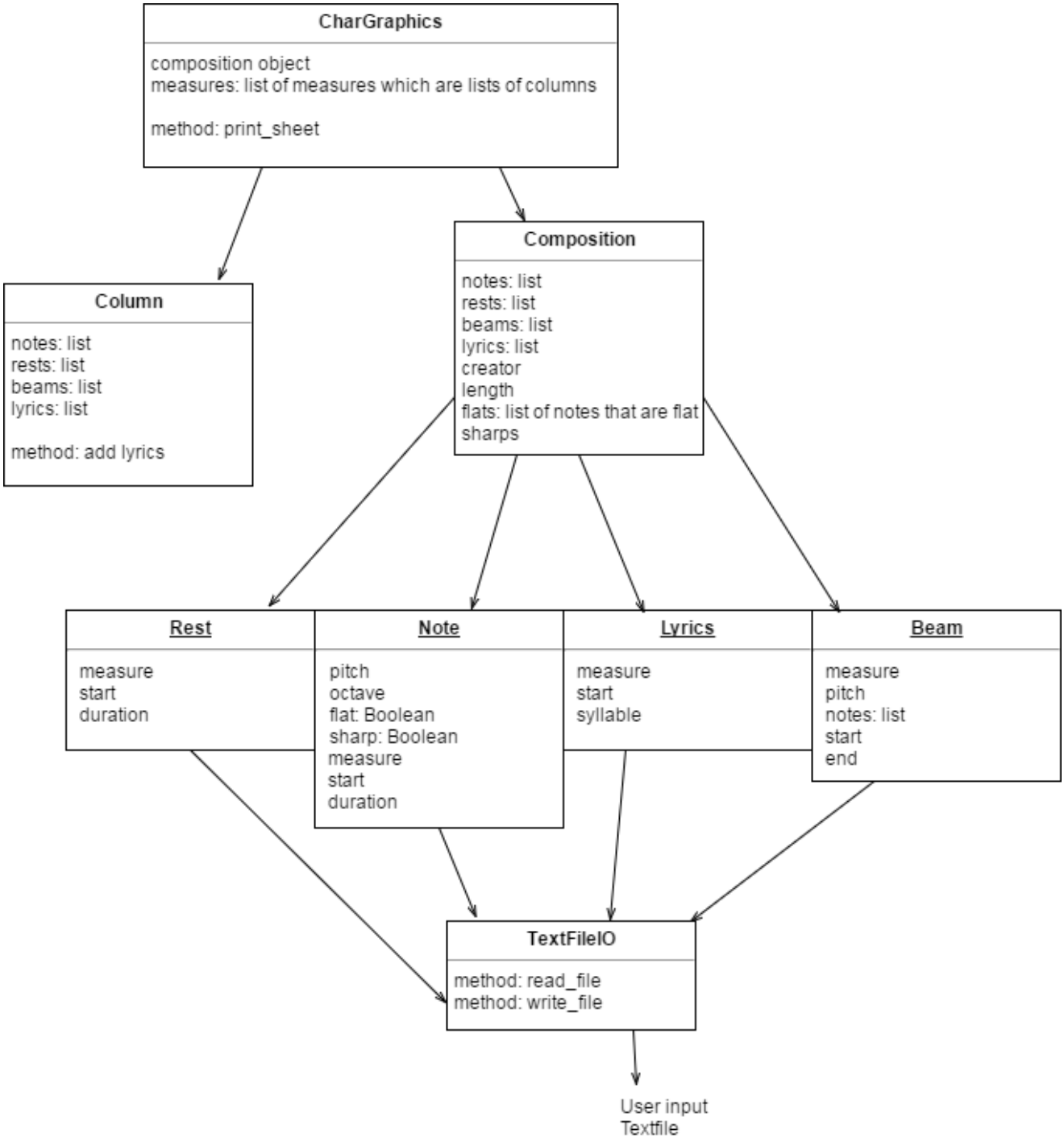
Add_measures() lisää measures-listaan sopivan määrän tahteja käyttäjän antaman pituuden mukaan.

Nuottiviivaston tulostus tapahtuu metodissa print_sheet. Metodissa create_measure lasketaan tarvittava tahdin koko, sen perusteella minkä pituisia ovat tahdin kestoltaan lyhyimmät tauot ja nuotit. Esimerkiksi jos lyhyin tauko/nuotti on pituudeltaan 1/16, luodaan tähän tahtiin 16 Column-oliota, jolloin tahti on 16*6 = 96 merkkiä pitkä. Tämä tekee nuottiviivastosta hyvännäköisen.

Luokassa Column piirretään 16*6-kokoiseen merkkimatriisiin tässä sarakkeessa olevat nuotit ynnä muut merkinnät. Lopulta on 4-ulotteinen lista, eli kaikki tahdit, jotka sisältävät tahdissa olevat kaksiulotteiset sarake-matriisit. Tämä matriisi piirretään metodissa prin_sheet luokassa CharGraphics käymällä läpi sisäkkäiset listat. Metodi tulostaa kolme tahtia kerrallaan, jotta nuottilehti rivittyy suurin piirtein sopivasti.

Metodi clef() piirtää nuottiavaimen ja alennetut sekä korotetut nuotit sen eteen.

Alla vielä karkea, notaatioltaan muotoa ’vapaa’, kaavio, joka havainnollistaa luokkien suhteita. Luokkien tärkeimmät attribuutit ja metodit on myös listattu:



5. Algoritmit

Luokassa **Beam**, metodissa add_pitch() oli hankaluutena määritellä mille korkeudelle piirretään palkki, johon liittyy on usein monella eri korkeudella olevia nuotteja. Metodissa add_pitch on nyt kolme vaihtoehtoa, jolla palkin korkeus (pitch) voi määrittyä. Jos mikään palkkiin liittyvistä nuoteista ei ole korkeampi kuin viivaston toisiksi ylin viiva, palkin korkeus on keskiarvo nuottien korkeuksista, joihin on lisätty 5 (nuotin varsi on yleensä 4 riviä korkea). Jos joku nuotti on toisiksi ylintä viivaa korkeammalla, mutta mikään nuotti ei ole korkeammalla kuin ylin F, palkki tulee ylimmälle riville eli ylimmän A-nuotin korkeudelle. Muuten palkki tulee alimmalle riville eli kääntyy

alaspäin. Käytin tässä algoritmissa Boolean-apumuuttujia. Metodi ei ole kommentoitu koodissa mutta tämä selitys varmaan avaa ideaa.

6. Tietorakenteet

Ohjelmassa käytetään paljon listoja tiedon varastoimiseen.

Esimerkiksi luokassa CharGraphics metodissa create_measure luodaan lista, jossa on kaikki yhden tahdin sarakkeet (Column-oliot). Column-oliot taas ovat 6*16-kokoisia kaksiulotteisia merkkilistoja. Jos Column-olio olisi esimerkiksi 16-pituinen lista kuusimerkkisiä muuttumattomia merkkijonoja, olisi mm. korotusmerkkien lisääminen paljon vaikeampaa. Muuttuvatilaiset listat tekevät merkkigrafiikan käsittelyn helpoksi.

Lopulta CharGraphics luokassa on neliulotteinen lista merkkejä, josta grafiikka tulostetaan yksi merkki kerrallaan.

Composition-oliossa on varastoitu kaikki sävellykseen sisältyvät nuotti-, tauko-, palkki- ja sanoitustavuoliot omissa listoissaan. Ohjelman keskeinen osa on sävellyksen muuttaminen ja listoista on helppo poistaa ja lisätä näitä olioita.

7. Tiedostot

Ohjelma ottaa käyttäjältä vastaan tekstitiedoston, jossa on sävelmän tiedot. Tiedostossa tulee olla ensimmäisenä otsake, muodossa ”Savellys X.X tallennustiedosto”, jossa X.X voi olla mikä tahansa merkkijono. Tiedostossa sävelmään käytettävät tiedot ovat lohkoissa #TIEDOT, #NUOTIT, #TAUOT, #PALKIT ja #SANOITUS. Lisäksi tiedostossa voi olla lohko #KOMMENTIT, joka ei sisällä piirrettävään sävelmään vaikuttavia tietoja, mutta joka tallentuu kun tiedostoon tehdään muutoksia. Teksti joka ei ole näissä lohkoissa katoaa kun ohjelma tallentaa tiedoston (eli käytännössä kirjoittaa tiedoston uudestaan).

Lohkot voivat olla keskenään missä järjestyksessä vain, ja tiedot lohkon tiedot voivat olla missä järjestyksessä vain.
Lohkojen syntaksi:

tiedot:
esim:

nimi: Yesterday
tekija: The Beatles
pituus: 7 [tahtien määrä]
tahtilaji: 4/4 [tämä jäi ajan puutteen takia toteuttamatta mutta jäi ohjelmaan]
alennetut: B
korotetut: C, A

nuotit:
<korkeus, oktaavi, alennus, ylennys, tahti, aloitushetki, kesto>
korkeus voi olla {A,B,C,D,E,F,G}
oktaavi voi olla {0,1} tai nuotille A myös 2
korotus ja alennus voi olla {e,k}

tahti voi olla {0,1,2,3,...,tiedot lohkossa annettu pituus-1}
aloitushetki voi olla esim. 4/8 riippuen mikä on tahdin lyhyin nuotti tai tauko. Jos tahdin pienin tauko tai nuotti on 1/8, tahdissa on vain kohdat 1/8, 2/8,...,7/8,8/8.
kesto voi olla {1, 1/2, 1/4, 1/8, 1/16} tai näistä puolikkaan keston verran pidennettyjä, esim. 3/2.
esim:
A, 1, k, e, 1, 3/8, 1/8

tauot:
<tahti, aloitushetki, kesto>
esim:
1, 1/8, 1/4

palkit:
<tahti, jarjestysnumero nuoteille aloitushetken ja korkeuden mukaan, erotetaan kaksoispisteellä>
esim:
1, 0:1:3

sanoitus:
<tahti, aloitushetki, tavu>
esim:
0, 1/8, Yes

Nuoteissa, tauoissa, palkeissa ja sanoituksissa on siis useita tällaisia rivejä. Repositoryssa src-kansiossa olevista tekstitiedostoista voi katsoa täysimittaisia esimerkkejä; yesterday.txt on oikea sävelmä johon on myös tavallinen nuottilehti kuvamuotoisena mallina kansiossa doc ja testi.txt – tiedostossa on erilaisia nuotteja ja taukoja mallina.

8. Testaus

Tein ohjelmalle järjestelmätestausta erilaisilla tekstitiedostoilla.

9. Tunnetut puutteet ja viat

Yksikkötestaus jäi ajan puutteen vuoksi kokonaan tekemättä. Metodeja olisi voinut testata esimerkiksi luokasta column määräämällä haluttuja palautusarvoja metodeille. Lisäksi luokan TextFileIO tekstinparsimiseen tarkoitettuja metodeja olisi pitänyt testata paljon erilaisilla virheellisillä ja myös virheettömillä inputeilla. Tähän luokkaan jäi varmasti puutteita ja luokka on muutenkin nyt sekava.

Erilaisten merkintöjen piirtämiseen olisi vielä paljon mahdollisuuksia. Tässä muutamia seuraavia askelia, joita ottaisin, jos jatkaisin ohjelman kehittämistä:

Nuottien ja taukojen kestoksi 1/32. Tämä onnistuu helposti samaan tapaan kuin muutkin kestot.

Tahtirajat ylittävät nuotit ja viivat (engl. tie) nuottien välille merkitsemään nuotin jatkumista. Tämä onnistuisi suurin piirtein samanlaisella luokalla kuin Beam, melko helposti.

Palkit ovat nyt vain 1/8-kestoisille nuoteille, eli yksinkertaisia. Lisäisin kaksinkertaiset palkit 1/16-kestoisille nuoteille.

Nuotin varsien piirtämisessä ilmenee välillä ongelmia, kun varteen pitäisi liittää monia nuotteja, yleensä kuitenkin toimii hyvin.
Sanoituksen lisääminen ja poistaminen samaan tapaan kuin nuottien ynnä muiden.

Lisää älykkyyttä esimerkiksi ilmoittamaan ongelmasta jos käyttäjä tekee musiikillisesti järjettömiä sävellyksiä. Esimerkiksi jos käyttäjä lisää heti nuotin jälkeen tauon, ennen kuin nuotin kesto on loppunut. (Ohjelmassa on jo algoritmi, joka poistaa tauon tai tekee tauoista oikean pituisia, jos heti tauon jälkeen tulee nuotti tai jos käyttäjä lisää uuden tauon vanhan eteen tai päälle. Katso metodit add_note ja add_rest luokassa composition.)

Jos käyttäjä lisää samalle kohdalle eripituisia nuotteja, ohjelma ei osaa erotella näitä, ja varret voivat näyttää väärältä.

Ohjelma ei osaa käsitellä vääränlaisia kestoja nuoteille ja tauoille. Jos tahdin pienin tauko tai nuotti on 1/8, tahdissa on vain kohdat 1/8, 2/8,...,7/8,8/8. Jos tauon tai nuotin aloitushetki poikkeaa näistä, ohjelma ei piirrä sitä.

10. Parhaat ja heikot kohdat

Heikkoja kohtia ohjelmassa on se, että on varmasti vielä paljon käyttäjä-inputteja joilla ohjelman saa kaatumaan.

Toiseksi, testaus-luokka jäi kokonaan toteuttamatta ajankäytöllisistä syistä.

Kolmantena heikkona kohtana mainitsen luokan TextFileIO, jossa on try-except rakenteita joista osa jäi turhaksi, ja muutenkin try-except rakenteet olisi voinut toteuttaa paremmin.

Parhaita kohtia ohjelmassa on luokka Beam, joka tekee palkkeja nuottien välille. Palkkeja ei eksplisiittisesti pyydetty tehtävänannossa, vaikkakin esimerkkikuvassa oli kuva nuoteista, joissa on palkkeja. Palkkeja voi lisätä useamman nuotin välille.

Toinen hyvä kohta ohjelmassa on luokka CharGraphics, joka piirtää kätevästi nuottiviivaston ja siinä metodi clef, joka piirtää myös nuottiavaimen (tämäkin tehtävänannon ulkopuolella).

Kolmantena huomioisin yleisesti sen, että ohjelmalla on sen verran erilaisia ominaisuuksia, että sillä voi piirtää esimerkiksi kappaleesta Yesterday muutaman tahdin melko helposti sekä oikeaoppisesti.

11. Poikkeumat suunnitelmasta

Suunnitelma oli melko, tai hyvin, suppea ja vain vähän mietitty, joten siihen tuli paljon muutoksia. Luokkia tuli uusia Beam ja Column. Kävi myös nopeasti selväksi ettei python tarvitse valmiita kirjastoja tämäntyylisen merkkigrafiikan tulostamiseen ja käsittelyyn, vaan se käy melko helposti listojen avulla.

12. Toteutunut työjärjestys ja aikataulu

Aloitin ohjelmoimisen huhtikuun kymmenennen päivän paikkeilla 2017, ja ensimmäisenä aloin luoda tulostuksen muotoilua kuten suunnittelinkin. Tässä meni noin kymmenen päivää ennen kuin aloin miettiä käyttäjän antaman tekstin parsimista, 22.4.2017. Sen jälkeen viimeistelin ohjelmaa kokonaisvaltaisesti.

Jos jakaa ohjelman toteutuksen karkeasti kahteen osaan, tulostukseen ja tekstin parsimiseen, tulostuksen toteutukseen meni arviolta 75% ajasta.

Kokonaisarvio projektin työmäärästä on todella karkeasti noin 60 tunnin luokkaa.

13. Arvio lopputuloksesta

Ohjelmassa on mielestäni riittävästi erilaisia ominaisuuksia ja nuottimerkintöjä, joita on mahdollista piirtää. Ohjelma tulostaa graafisesti ja esteettisesti toimivia nuottiviivastoja, jotka tunnistaa nuottiviivastoiksi. Nuottien ym. lisääminen ja poistaminen onnistuu melko luontevasti käyttöliittymän avulla, ja ohjelma tallentaa tiedostoja joita se osaa itsekin lukea.

Koska omat ohjelmointitaidot ovat rajalliset ja kokemus melko vähäistä, koodin tyylissä on varmasti parantamisen varaa. Luokat Note, Rest, Beam ja Lyrics voisivat varmaankin olla aliluokkia yhdestä luokasta (esim. Item), koska niillä on muutamia samoja attribuutteja, ja niitä käytetään ohjelmassa hyvin samalla tavalla.

Merkkien käsittely listoissa mahdollistaisi ohjelman helpon laajentamisen. Jos merkit olisi muuttanut merkkijonoiksi liian aikaisin ennen tulostamista, olisi ohjelma jäykempi ja vaikea laajentaa.

Ohjelmaan voisi lisätä ominaisuuksia vielä todella paljon, koska nuottimerkintöjä on paljon ja notaatiot moninaiset.

14. Viitteet

Etsin tietoa netistä, lähinnä:

https://stackoverflow.com
https://docs.python.org

myös nuottinotaatioiden oppimiseen tarvitsin apua, ja apua löytyi:

http://www.musicnotes.com/blog/2014/04/11/how-to-read-sheet-music/

15. Liitteet

En osaa käyttää scriptejä, mutta käyttöohjeet-osiossa on yksi esimerkkikomentojono.

Repositoryssä on kuva mallina toimineesta The Beatlesin kappaleen nuoteista ja kuva siitä mitä ohjelman pitäisi tulostaa scr-kansiossa olevan yesterday.txt-filun avulla.