BSP3 project: Computational Tools for Music Python Conversion

Generated by Doxygen 1.9.2

Chapter 1

BSP3

PYTHON CONVERSION README BSP3 project: Computational Tools for Music Python Conversion

This Readme file concerns the python translated version of a java project about the musical structures discussed by Luigi Verdi in his book "Caleidocicli Musicali". This project was developped by a student at the University of Luxmebourg under the supervision of M.di Tollo. To understand all the key terms of the project, please refer to the BSP report written alongside the code.

PLEASE NOTICE that this software has been developed to be used on Windows and portability is not guaranteed.

Required language: Python 3.10.0 Installation process: Follow the link https://www.python.↩org/downloads/ and select the version 3.10.0 for the OS that you are using. Since this project was developped on Windows 10 so using this OS is recommended. Once the download started, follow the instrcutions given on the installer. For additional information for Windows follow this link: https://phoenixnap.↩com/kb/how-to-install-python-3-windows To install on Linux, follow this link: https↩://opensource.com/article/20/4/install-python-linux To install on Mac OS, follow this link: https://www.freecodecamp.org/news/python-version-on-mac-update/

Required libraries: Pip, PyQt5, Qt Designer, pyqtgraph Installation processes for Windows: Pip: Pip is a package manage for python. Make sure python is installed before continuing. The main documentation is found here: https://pypi.org/project/pip/ Installation process is found here: https://phoenixnap.↩com/kb/install-pip-windows

PyQt5: The PyQt5 is a module that is used to create desktop applications in python. Make sure pip is installed before continuing. The installation process is found here: https://pythonbasics.org/install-pyqt/

Qt Designer: Qt Designer is used to make the PyQt5 application designing process easier. Make sure PyQt5 is installed before continuing. The installation process is found here: https://build-system.fman.↩io/qt-designer-download

pyqtgraph: pyqtgraph is used to design graphs for the visual representations The installation process is found here: https://pyqtgraph.readthedocs.io/en/latest/installation.html

HOW TO RUN THE PROGRAM: Make sure you are in the correct path: ...\LUIGIVERDI\Proj3_PythonConvertion To run the code in the terminal use the command: python main.py Three questions will appear and ask the different parameters. If the parameters entered are not valid, the code will not run. There are 5 tabs: -the first one gives general information about the cycle, the two kaleidos and the two kaleidocycles -the four other tabs are the visual representation of the objects mentionned above The user can zoom in/out as well as scoll to the sides on the graphs, but this is not necessary since the graphs are scaled automatically to see all the points in the window.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Cycle Namespace Reference

Classes

- class Cycle

5.1.1 Detailed Description

File containing the cycle class, needed for all other classes

5.2 Kaleidocycle Namespace Reference

Classes

- class Kaleidocycle

5.2.1 Detailed Description

File containing the kaleidocycle class

5.3 KaleidocycleTranspose Namespace Reference

Classes

- class KaleidocycleTranspose

5.3.1 Detailed Description

File containing the kaleidocycle transpose class

5.4 Kaleidos Namespace Reference

Classes

- class Kaleidos

5.4.1 Detailed Description

File containing the kaleidos class

5.5 KaleidosTranspose Namespace Reference

Classes

- class KaleidosTranspose

5.5.1 Detailed Description

File containing the kaleidos transpose class

Chapter 6

Class Documentation

6.1 Cycle.Cycle Class Reference

Public Member Functions

- def **__init__** (self, module, nb_notes)

Static Public Member Functions

- def composanteCalc (x, meter, composante, base, notes, cycle)
- def cycleCalc (f, meter, composante, base, notes, cycle)
- def phaseCalc (cycle)

Public Attributes

- **module**
- **nb_notes**
- **meter**
- **base**

Static Public Attributes

- int **meter** = 0
- int **base** = 0
- list **cycleSet** = [ ]
- list **composante** = [ ]
- int **phase** = 0

6.1.1 Detailed Description

```
a cycle is composed of a: meter, base, composante, phase and noteSet
```

6.1.2 Member Function Documentation

6.1.2.1 composanteCalc()

```
def Cycle.Cycle.composanteCalc (
              x,
              meter,
              composante,
              base,
              notes,
              cycle )  [static]
```

function calculates the first composante when called for the first time, then calculates the other ones

6.1.2.2 cycleCalc()

```
def Cycle.Cycle.cycleCalc (
              f,
              meter,
              composante,
              base,
              notes,
              cycle )  [static]
```

function calculates all the elements of the composante until the last element of a composante is equal to 0

6.1.2.3 phaseCalc()

```
def Cycle.Cycle.phaseCalc (
              cycle )  [static]
```

function calculates the amounts of phases (more practical than always using len(cycle.cycleSet))

The documentation for this class was generated from the following file:

- Cycle.py

6.2 Kaleidocycle.Kaleidocycle Class Reference

Public Member Functions

- def __init__ (self, cycle, kaleidos, nb_notes)

Public Attributes

- **cycle**
- **kaleidos**
- **nb_notes**

Static Public Attributes

- list **composante** = [ ]

6.2.1 Detailed Description

```
a Kaleidocycle only has a composante
```

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

```
def Kaleidocycle.Kaleidocycle.__init__ (
              self,
              cycle,
              kaleidos,
              nb_notes )
```

```
in this function, the composante is calculated
Each element i of the Kaleidocycle is made of the "Kaleidos-i" element of the cycle
```

The documentation for this class was generated from the following file:

- Kaleidocycle.py

6.3 KaleidocycleTranspose.KaleidocycleTranspose Class Reference

Public Member Functions

- def __init__ (self, kaleidocycle, k, nb_notes)

Public Attributes

- **kaleidocycle**
- **k**
- **nb_notes**

Static Public Attributes

- list **composanteT** = [ ]

6.3.1 Detailed Description

```
a kaleidocycle transpose only has a composante transpose and uses the previously calculated kaleidocycle
```

6.3.2 Constructor & Destructor Documentation

6.3.2.1 __init__()

```
def KaleidocycleTranspose.KaleidocycleTranspose.__init__ (
              self,
              kaleidocycle,
              k,
              nb_notes )
```

```
by using the k value, add it to every value of the composante to obtain the transposed version of the kaleidoc
```

The documentation for this class was generated from the following file:

- KaleidocycleTranspose.py

6.4 Kaleidos.Kaleidos Class Reference

Public Member Functions

- def **__init__** (self, cycle, nb_notes)

Static Public Member Functions

- def composanteCalc (cycle, kaleidos)

Public Attributes

- **cycle**
- **nb_notes**

Static Public Attributes

- list **accessory** = [ ]
- list **structVert** = [ ]
- list **composante** = [ ]

6.4.1 Detailed Description

```
a Kaleidos is composed of an accessory, a vertical structure and a composante
it also requires a Cycle object to be created
```

6.4.2 Member Function Documentation

6.4.2.1 composanteCalc()

```
def Kaleidos.Kaleidos.composanteCalc (
              cycle,
              kaleidos )  [static]
```

```
function calculates the elemnts of the composantes by going through the elements of the vertical structure
```

The documentation for this class was generated from the following file:

- Kaleidos.py

6.5 KaleidosTranspose.KaleidosTranspose Class Reference

Public Member Functions

- def __init__ (self, kaleidos, k, nb_notes)

Public Attributes

- **kaleidos**
- **k**
- **nb_notes**

Static Public Attributes

- list **structVertT** = [ ]

6.5.1 Detailed Description

kaleidos transpose class only has a vertical structure and uses the kaleidos calculated previoulsy

6.5.2 Constructor & Destructor Documentation

6.5.2.1 __init__()

```
def KaleidosTranspose.KaleidosTranspose.__init__ (
              self,
              kaleidos,
              k,
              nb_notes )
```

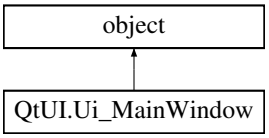
by using the k value, add it to every value of the vertical structure to obtain the transposed version of the

The documentation for this class was generated from the following file:

- KaleidosTranspose.py

6.6 QtUI.Ui_MainWindow Class Reference

Inheritance diagram for QtUI.Ui_MainWindow:



Public Member Functions

- def setupUi (self, MainWindow)
- def retranslateUi (self, MainWindow)

Public Attributes

- **centralwidget**
- **tabWidget**
- **tab**
- **label**
- **tab_2**
- **graphicsView**
- **tab_3**
- **tab_4**
- **tab_5**
- **statusbar**

6.6.1 Detailed Description

```
Main window is composed of 2 functions to setup the different components of the UI as well as the data that th
and to add the various tabs to the tabWidget (including titles and some label text)
```

6.6.2 Member Function Documentation

6.6.2.1 retranslateUi()

```
def QtUI.Ui_MainWindow.retranslateUi (
              self,
              MainWindow )
```

```
This function adds the different tabs to the window and also adds the information required on the information
```

6.6.2.2 setupUi()

```
def QtUI.Ui_MainWindow.setupUi (
              self,
              MainWindow )
```

```
this function creates the different tabs that are in the main window and also adds the graphs containing the v
of the kaleidos/kaleidosT and kaleidocycle/kaleidocycleT
All the sizing, fonts etc... are done here
```

The documentation for this class was generated from the following file:

- QtUI.py

Index