To know before extending the BIP code: basic Python and OOP knowledge (inheritance, decorators, optimizing…)

For more detailed explanations about the code, please refer to the code itself and its commentaries.

The project structure:

The only two elements that blender will execute is main.py and run.py. These files use all other files to create, interfacing, and manage all elements of the presentation.

Our code is composed by 4 main elements: the *Interface elements*, the *Managers elements*, the *Models elements*, and the *running* elements.

Interface elements:

This module will register all menu-related elements to help created the presentation. There are three elements in it, the Menus one which represents the submenus elements. The Operators one that will represent all operators elements (button mostly). And the Props elements that will represents all properties elements (like slide number for example).

All Menu objects need to be inherited from the Menu interface class (see Menus.py for more information) as operators objects and props objects.

All classes created in these files need to be registered in the classes structure on main.py

Manager elements:

Manager's elements are the managers of the project. It contains AnimationManager, SlideManager, and XMLParser.

The AnimationManager is a static class that contain all possible annimations. You can create another animation type by adding a method in it and linking it on the animation(…) method.

The SlidesManager contain two elements: the Slides class and the SlidesManager class. The Slides class contain every element necessary to a slide (its position, its objects…) The SlideManager is a Singleton (see Singleton part for more details) whish contain all slides, the current slide position, the camera…

The XMLParser is the parser of the XML translation file. You don't need to edit this file to create another translation. All you need is to change his call using init_XMLParser.

Usage: Init_XMLParser(path_to_xml, language_subnode, [tab_of_all_elements_to_get])

For example: getting the English subnode of the texts.xml file whish contains only labels and button elements:

```
<?xml version="1.0" encoding="UTF-8"?>
<TextData version="TDv1">
  <en-US>
    <label  type="Element1">First element</label>
    <button  type="Element2">Second element</label>
  </en-US>
</TextData>
```

we will do:
init_XMLParser(path("texts.xml"), "en-US", ["label", "buttons"])

And we get the translation using the XMLParser by doing:
XMLData["type@element"]

Example: for getting the element "First element": we will do:
First = XMLData["label@Element1"]

Model Element:

The model element only contain the element Singleton whish is a decorator to use on a class for creating a singleton. A decorator is a Python function that has the possibility to modify another function of a class. This singleton decorator will take the decorated class in parameter and add to it all needed elements to create a Python Singleton Class. For more information, please refer to the code.

Running elements:

The running elements are ***main.py*** and ***run.py***. main.py create the interface for creating the presentation. And run.py execute the presentation.

On ***main.py*** you can find a structure called classes, in whish one we need to set all created interface classes for Blender.