

## **CO32016 Group Project 3**

# **Integration of Lip-Synching in Blender3D**



Group Members:

- **Aina Rakotomalala** (06012521)
- **Bamar Diagne** (06012269)
- **Daniel Lovera** (06012373)
- **Marc Gardent** (06012380)

Supervisor: **Dr. Xiaodong Liu**  
2006 – 2007

Napier University  
School of Computing

## Table of content

<b>I. PROJECT DEFINITION .....</b>	<b>4</b>
<b>1.1 Composition of the group .....</b>	<b>4</b>
<b>1.2 Project scope .....</b>	<b>4</b>
1.3 Project aims .....	5
1.4 Tools and technologies used .....	6
1.5 Risks .....	6
<b>II. PROJECT PLAN .....</b>	<b>7</b>
2.1 Gantt chart .....	7
2.2 Risk and contingency .....	7
<b>III. REQUIREMENTS ANALYSIS .....</b>	<b>9</b>
<b>3.1 User Identification .....</b>	<b>9</b>
<b>3.2 User Requirements .....</b>	<b>10</b>
3.2.1 Functional Requirements .....	10
3.2.2 Technical Requirements .....	16
3.2.3 Appearance Requirements .....	17
3.2.4 Usability and Performance Requirements .....	17
Usability requirements .....	17
Speed and latency requirements .....	17
Precision or Accuracy Requirements .....	17
3.2.5 Release and legal Requirements .....	18
Release Requirements .....	18
Products That Can Be Reused .....	18
<b>3.3 Context of use .....</b>	<b>19</b>
3.3.1 Target Audience .....	19
3.3.2 Locations and Time of usage .....	19
3.3.3 Reasons of usage .....	19
3.3.4 Usability .....	20
<b>IV. ACCEPTANCE TESTS .....</b>	<b>25</b>
<b>4.1 User Requirements .....</b>	<b>25</b>
4.1.1 Functional tests .....	25
4.1.2 Appearance tests .....	36
4.1.3 Usability and Performance tests .....	37
Usability tests .....	37
Speed and latency test .....	39
4.1.4 Release and legal test .....	40
Release tests .....	40

<b><i>V. DESIGN DESCRIPTION</i></b> .....	<b>42</b>
<b><i>5.1 Use-Cases</i></b> .....	<b>42</b>
<b><i>5.2 UML Class chart:</i></b> .....	<b>46</b>
<b><i>5.3 UML sequence</i></b> .....	<b>47</b>
<b><i>5.4 Implementation</i></b> .....	<b>49</b>
<b><i>VI. USER MANUAL</i></b> .....	<b>50</b>
<b>6.1 Introduction: Welcome</b> .....	<b>50</b>
<b>6.2 Open the Lip-Sync Editor</b> .....	<b>51</b>
<b>6.3 Create/Select a lip-sync project</b> .....	<b>52</b>
<b>6.4 Load an audio file</b> .....	<b>52</b>
<b>6.5 Create/Load a text file</b> .....	<b>52</b>
<b>6.6 Adjust timing by moving sentences, words and phonemes</b> .....	<b>53</b>
<b>6.7 Reload text and sound used</b> .....	<b>53</b>
<b>6.8 Set the starting frame of your lip-sync project</b> .....	<b>54</b>
<b>6.9 Create/Select a mapping project</b> .....	<b>54</b>
<b>6.10 Link the phonemes to one or several shape keys</b> .....	<b>55</b>
<b>6.11 Create a new phoneme and delete a created phoneme</b> .....	<b>55</b>
<b>6.12 Bake animation</b> .....	<b>56</b>
<b>6.13 Your animation is ready</b> .....	<b>56</b>
<b><i>VII. GLOSSARY</i></b> .....	<b>57</b>

# *I. Project Definition*

## **1.1 Composition of the group**

Group number: 3

Supervisor: Dr. Xiaodong Liu

- 06012521, Aina Rakotomalala
- 06012269, Bamar Diagne
- 06012373, Daniel Lovera
- 06012380, Marc Gardent

To get more information about 3D and animation we have decided to contact Dr. Gregory Leplatre. Indeed, 3D is the main topic of our project and he teaches 3D at the university. We met him to know his point of view as an animator.

## **1.2 Project scope**

3D is very developed and people can see 3D everywhere: cinema, advertisements, cartoons, effects of movies. Therefore, more and more people are interested by 3D and more and more software are available to create 3D characters and 3D universes. Some free open-source 3D software are available on internet. Now, with training, people are able to create everything with 3D.

Usually, 3D means realism and life. Moreover, people have to create an animation of the character to make it live. Animations are not easy to make and need much time to be done. For our project, we only focus on the lips' animations. All animations where one or several characters have to speak need to have a *lips synchronisation*, called *lip sync* or *lip-sync* too. The lip sync takes a lot of time because the animator has to match each syllables said by the character with an accurate lips animation.

As this year, all 3<sup>rd</sup> year students have to do a group project, we saw the opportunity to develop something interesting, ambitious and especially useful for 3D animators.

Therefore, the time left for us by the university is 12 weeks to think to plan and to design the project, to think about all the technical requirements and to develop the project. Each step will be developed during the project and will be noted in details in our logbooks and in this document. Consequently, these parts will be to develop in this document:

- A **project plan** to split in detail each task of the project and to allocate the human resources.

- A **requirements analysis** to determine the essential requirements of the system, taking account of the possibly conflicting requirements of the various users.
- **Acceptance tests** to prove the user that all requirements are satisfied.
- A **design description** to explain the structure of our project.
- A **user manual** to help the user to use our interface.

### 1.3 Project aims

Our project has as aim to integrate a new interface, inspired by software such as Papagayo, Yolo or Jlip-sync, in the 3D software Blender. This new and friendly interface allows doing the lip sync process more easily by opening an audio file and by a text input. Then the user used our interface into the 3D software Blender to process to the lip sync. After that the user has finished synchronising all his shapes keys, the animation of the lips is finished and the user can export it to a video. Our interface does not export sounds to the video. Indeed, the user will have to merge the video exported and the sounds with a video editing software.

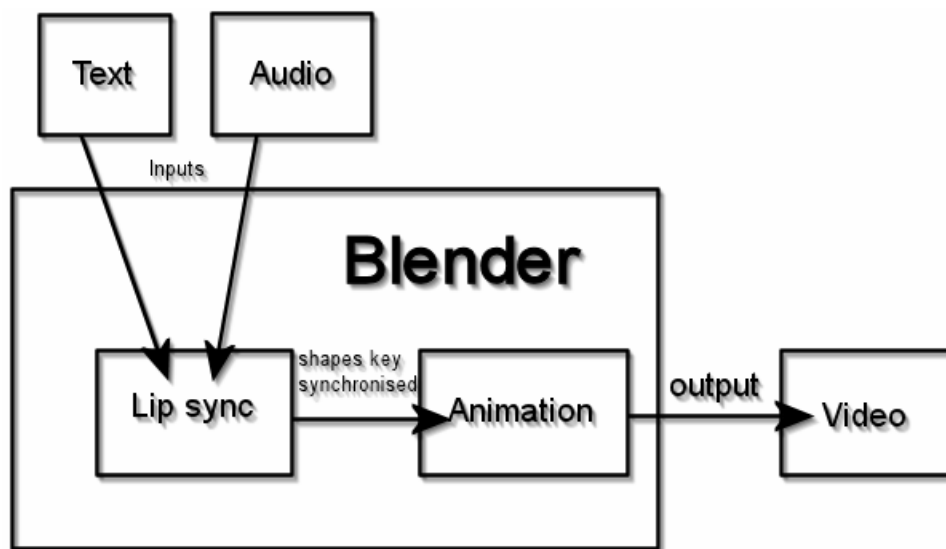


Illustration 1: General process

The other aim of this project is to learn how to work in-group. This means to learn to split wisely each task of the project and to allocate to each member of the group according to their skills, their experience and their interests. Indeed, usually in a company, people work in a little group or a bigger group. Understanding the group spirit, adapting to the other members or finding a compromise during a conflict are the main things to know to manage well a group project.

### ***1.4 Tools and technologies used***

For our project, we will develop an interface on the 3D software Blender to apply the lip sync on 3D character. Therefore, we need the open-source code of Blender. Many compilers are available to compile Blender and be able to work on the source code, such as Cygwin, Dev C++, Scons or MS Visual Studio. We have chosen to work with MS Visual Studio because most of us have already installed it for previous coursework, so we have a good knowledge about the environment. Although Blender can run on all platforms, we mainly use Microsoft Windows XP as operating system to make the project. However, we also want to make our project run on all platforms too, that's why we are using Scons on the Linux platform.

In addition, we use the open-source code of Papagayo (in Python language) that we translate to C++ in order to integrate the lip sync function of Papagayo in Blender. We have created a wiki (<http://wiki.blender.org/index.php/Requests/lipSync>) about our project in order to communicate the project to the Blender community and to get subjections from users and experts. This is really useful especially when we need help or advices to improve the interface and to keep the "Blender's spirit".

### ***1.5 Risks***

Several risks can appear during the realisation of this project:

- **Time constraint** (Risk: High): Our project is related highly with the 3D software Blender and its code is hard to read and to understand. We need time to understand the code and to implement it. In addition, we have to do a compromise between time and quality of our interface.
- **Lack of skills** (Risk: High): As Blender is programmed by high skilled people, we may have a lack of skills at the beginning to understand the code.
- **Installing software** (Risk: Medium): We only need of Microsoft Visual Studio 2002 or 2003 and the source code of Blender. As everybody have Microsoft Visual Studio 2003 thanks to Microsoft Alliance, the main difficulty is to compile Blender. Although tutorials about compiling Blender exist on internet, compiling Blender requires time.
- **Requirements change** (Risk: Medium): Requirement can change after the suggestions give by the Blender community. These suggestions can have an impact on the development of our project. It would be better to get their advices before the beginning of the development instead of during the development because it is difficult to change the design of the implementation during the development.

## *II. Project Plan*

### **2.1 Gantt chart**

The Gantt chart may appear not optimized at some points, considering that most of the tasks could be performed successively instead of delaying. The reason is we keep in mind the amount of work we have to produce for the other courseworks. Omitting this parameter could be damaging for the good realization of the project, our efficiency could be really reduced when it is not considered properly.

This following Gantt chart presents you the steps to be undertaken, their order, the people who will work on them, the estimated duration and effort required for each phase.

### **2.2 Risk and contingency**

- Programming environment: the compilation under MSVC++ doesn't work so we can install Cygwin (Linux-like environment for Windows).
- Coding: The cloning of the audio module step is a crucial one because, it's achievement is required by many others programming steps. If we spend too much time to realize it, it can compromise the well going of the project.
- Certain steps of our application require a minimum of functionality from the Blender modules. In that case we assume that the elements we are reusing works fine. But bugs are discovered from time to time, so some of them might corrupt our process even if they are external. For example the "baking" step requires many external modules so can be risky.
- Deployment: Because of the nature of the work, embedded in a world wild project. We have several expectations to respect in order to be included in the CVS (Current Version System) of Blender. That means many procedural steps to achieve that goal. So we'll probably have to make adjustment required by the blender development community. If we don't have enough time to realize that properly, we could anyway submit our software as a patch, and because of it's "open source" nature others if not us can keep the work going.

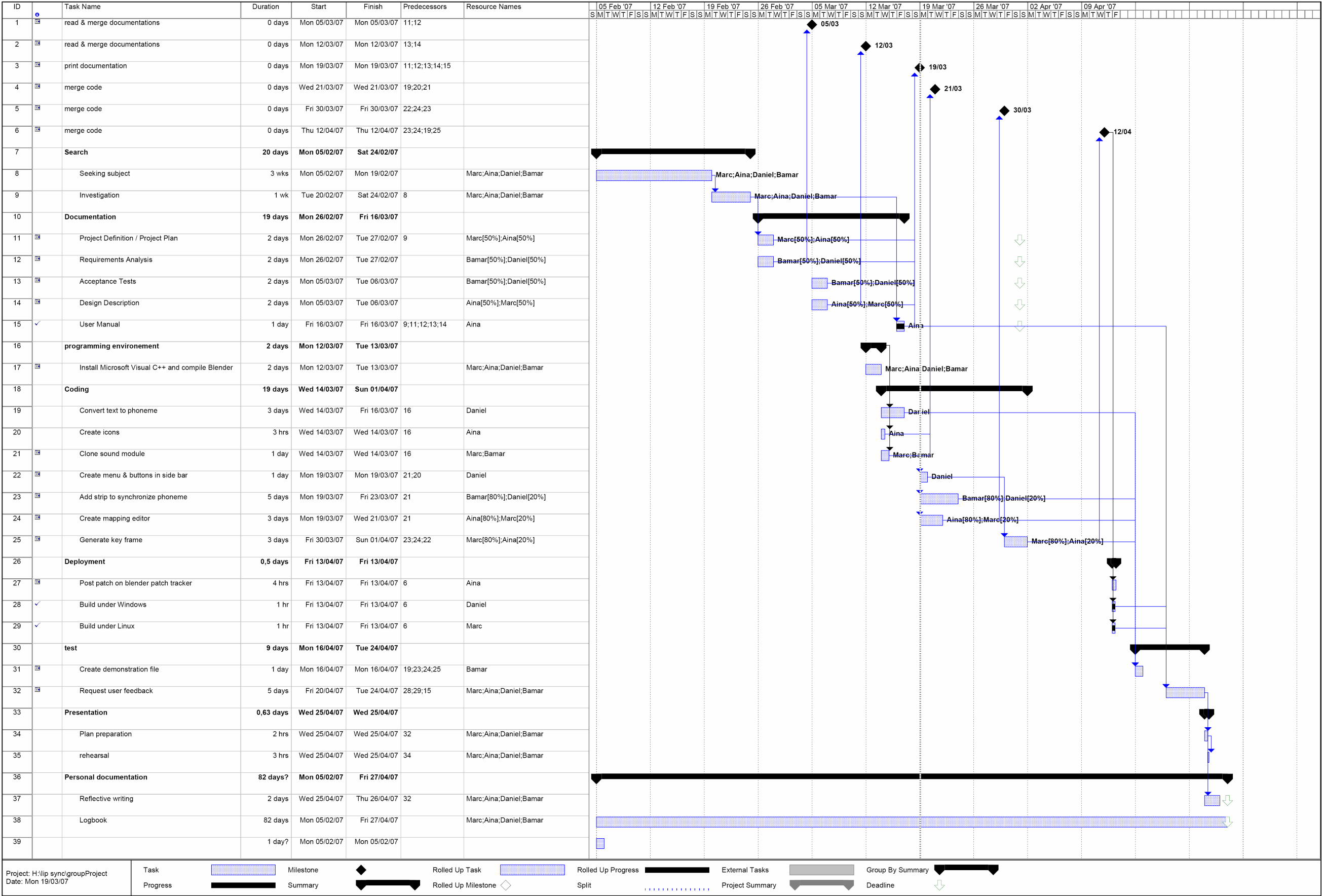


Illustration 2: Project Plan



### *III. Requirements Analysis*

Communicating with users will allow us to know about user requirements so we could define our system requirements. In order to determine what user requirements are we need to define a user profile for our project.

#### **3.1 User Identification**

As our project is to integrate a new animation feature in an existing 3D software, one of the aim of our project is to fit expectations of people using this software and especially animators. Lip synchronization can be a technical step in 3D animation and needs knowledge to be set up but it has to be done outside the software used to produce 3D modelling. People doing lip synchronization in 3D can be animators working on a scene (such as in a cinema or video game realisation), scientist doing research experimentation, novice user experimenting animation (such as students), etc.

Indeed, Blender is a free software, and often used for non professional work so we need to keep in mind that our integrated feature must be flexible enough to fit professional animators expectations but also novices expectations. Another kind of user that we need to consider are test engineers and people developing features in blender. Indeed, even if we will have to produce our own tests according to some blender development requirements, some testers may check our code in order to know if it would be integrated without modification in the next blender release. In the same way, as Blender keeps evolving we need to consider people that may want to upgrade our system such as Blender developers.

For some category of user, we need to attach a priority in order to know which requirements we will have to focus on in case of conflicts. Priority is given to a category of user following user knowledge and experience, time allocated using the system, and representative percentage on all users. So the more the category will fit the knowledge and experience required, spend time on our system, and represent a large percentage of all users, the more it will tend to be key users.

Key users: Animators.

Secondary users: Casual and non professional users.

It is important to note that test engineers and developers requirements won't be in conflicts with other users and need to be handle separately. So we can say that our main user requirements will be know by asking animators but some constraints will remain such as:

- A clear interface and a flexibility to allow non professional people to use parts of our application
- A way of coding, comment and debug the application for testers from the Blender Community.

### 3.2 User Requirements

As we know that communication with users could be slow, we have started our investigation since the earlier stages of our project. What we have done is to create a wiki page explaining our project and discuss technical issues and user requirements with Blender's lead technical programmers Ton Roosendaal and Tom Musgrove and an animator from the movie Elephant Dream that was in charge of some of the lip synchronization, Bassam Kurdali, on the "Blendercoders" IRC chat. We have also written topics on forums to receive general users' comments and suggestions about our project.

The wiki page has been changed several times and some user requirements and issues had appeared.

#### 3.2.1 Functional Requirements

##### System start-up

When the user starts the system, Lip sync editor and Mapping editor will be directly accessible.

##### List of requirements

Requirement: 1	
Event:	none
Description:	The system allows the user to create a "Lip sync project".
Rationale:	To be able to apply lip sync on a created object.
Fit Criterion:	The user selects an object and activates a control to create a lip sync project for this object.
Supporting materials:	Event list (3.2.1 ), use case diagram (3.2.1), usage step 2 (3.3.4)

Requirement: 2	
Event:	none
Description:	The system allows the user to select an existing "Lip sync project".
Rationale:	To be able to apply an existing lip sync on a created object.
Fit Criterion:	The user selects an object and selects a lip sync project in a list box for this object.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 2 (3.3.4)

Requirement: 3	
Event:	none
Description:	The system allows the user to close a "Lip sync project".
Rationale:	To be able to unlink a lip sync to a selected object.
Fit Criterion:	The user selects an object and activates a control to close its lip sync project.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 2 (3.3.4)

Requirement: 4	
Event:	1
Description:	The system allows the user to load a sound file.
Rationale:	To be able to use an audio recorded file as the voice of the object.
Fit Criterion:	The user browses its hard drive and chooses an audio file.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 4 (3.3.4)

Requirement: 5	
Event:	none
Description:	The system allows the user to "close" an used sound file.
Rationale:	To be able to unlink an audio file previously applied as the voice of an object.
Fit Criterion:	The user browses its hard drive and chooses an audio file.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 4 (3.3.4)

Requirement: 6	
Event:	2
Description:	The system allows the user to load a text.
Rationale:	To be able to use a text to synchronize with the audio.
Fit Criterion:	The user can browse his hard disk to load a text file or activate a control and launch the text editor. Indeed, the text editor will allow him to type a text or load an entire text file by browsing his hard drive.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 5 (3.3.4)

Requirement: 7	
Event:	none
Description:	The system allows the user to "close" an used text.
Rationale:	To be able to unlink a text previously linked to an object.
Fit Criterion:	The user selects an object and activates a control to close the text linked to this object.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 5 (3.3.4)

Requirement: 8	
Event:	none
Description:	The system allows the user to adjust timing and text with the audio in the editor.
Rationale:	To be able to synchronize text with the audio.
Fit Criterion:	The user resizes and drags & drops sentences, words, and phonemes in the editor so it becomes synchronized with the audio.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 6 (3.3.4)

Requirement: 9	
Event:	3
Description:	The system allows the user to reload the text and sound file used.
Rationale:	To be able to update the text and sound used for the lip sync when it has been modified.
Fit Criterion:	The user activates a control that will reload the content of the previous text (either a file or the content of the text editor) and the content of the previous sound file.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 5 (3.3.4)

Requirement: 10	
Event:	none
Description:	The system allows the user to set the starting frame of the "Lip sync project".
Rationale:	To be able to start lip sync on an object anytime during a scene.
Fit Criterion:	The user type the number of the starting frame he has decided in a text area.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 6 (3.3.4)

Requirement: 11	
Event:	none
Description:	The system allows the user to create a "Mapping project".
Rationale:	To be able to link phonemes to an object.
Fit Criterion:	The user selects an object and activates a control to create a mapping project for this object.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 3 (3.3.4)

Requirement: 12	
Event:	none
Description:	The system allows the user to select an existing "Mapping project".
Rationale:	To be able to link phoneme to an object according to an existing mapping.
Fit Criterion:	The user selects an object and selects a mapping project in a list box for this object.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 3 (3.3.4)

Requirement: 13	
Event:	none
Description:	The system allows the user to close a "Mapping project".
Rationale:	To be able to unlink a mapping project to a selected object.
Fit Criterion:	The user selects an object and activates a control to close its mapping project.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 3 (3.3.4)

Requirement: 14	
Event:	none
Description:	The system allows the user to link a "Lip sync project" to a "Mapping project".
Rationale:	To be able to use different mapping projects with a text and sound synchronized timing (a lip sync project).
Fit Criterion:	The user activates a control in the lip sync editor and chooses a mapping project to be linked with in a list box.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), (3.3.4)

Requirement: 15	
Event:	none
Description:	The system allows the user to create a new phoneme.
Rationale:	To be able to create new phonemes independently from the text.
Fit Criterion:	The user activates a control to create a new phoneme.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 8 (3.3.4)

Requirement: 16	
Event:	none
Description:	The system allows the user to select a phoneme.
Rationale:	To be able to link the selected phoneme to one or more shape keys.
Fit Criterion:	The user selects a phoneme in a list box.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 8 (3.3.4)

Requirement: 17	
Event:	none
Description:	The system allows the user to delete a customized phoneme.
Rationale:	To be able to delete customized phoneme created by user.
Fit Criterion:	The user selects the phoneme he wants to delete and activates a control to delete it.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 8 (3.3.4)

Requirement: 18	
Event:	4
Description:	The system allows the user to link each phoneme to a combination of several shapes.
Rationale:	To be able to produce a realistic animation according to the audio and text.
Fit Criterion:	The user activates a control corresponding to a phoneme and chooses a percentage of each shape he has created using scrolling buttons.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 8 (3.3.4)

Requirement: 19	
Event:	5
Description:	The system allows the user to “bake” his lip sync animation.
Rationale:	To be able to keep and reuse the animation created, and avoid the whole process of making animation to be restarted on each modification.
Fit Criterion:	The user activates a control to “bake” the combination of shapes linked to phonemes timing.
Supporting materials:	Event list (3.2.1), use case diagram (3.2.1), usage step 6 (3.3.4)

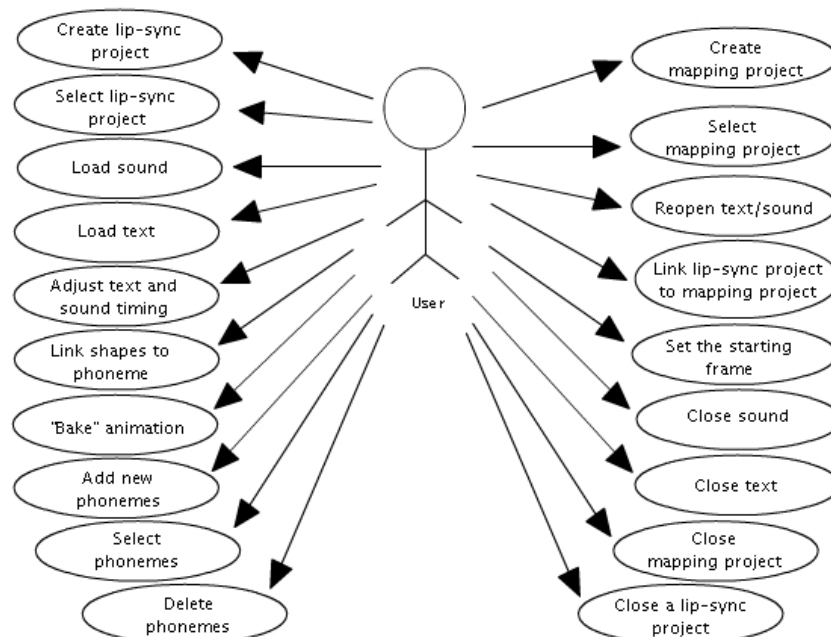
## Event List

The event list allows us to analyze some inputs from the user and outputs from our system.

Event name	Input and output	Summary
1. Load sound	.wav file (in)	Browse hard drive and select a .wav file.
2. Load text	.txt file or text editor content (in)	Browse hard drive and select a .txt file or use the text editor.
3. Reload text and sound	.txt file or text editor content (in) .wav file (in)	Update the content of the modified text and sound.
4. Link shapes to phonemes	Create shape keys combination according to the timing (out)	Link each phoneme to one shape or a percentage of each shape.
5. Bake animation	Animation (out)	Save the animation according to the settings

**Illustration 3: Event list of inputs and outputs**

## Use Case Diagram



**Illustration 4: Use-cases from the user's point of view**

### 3.2.2 Technical Requirements

To be able to run our tool properly the user has to match some hardware and software requirements. As we're developing by using the source code of Blender 2.43, his technical requirements apply for our tool as well. And a configuration near to the optimal settings provides to the user a better environment for animations processes.

#### Software requirements: Operating Systems

- Windows 98, ME, 2000, XP or Vista
- Mac OS X 10.2 and later
- Linux 2.2.5 i386
- Linux 2.3.2 PPC
- FreeBSD 6.2 i386
- Irix 6.5 mips3
- Solaris 2.8 sparc

#### Hardware requirements:

##### Minimal specifications for Hardware

- 300 MHz CPU
- 128 MB Ram
- 20 MB free hard disk Space
- 1024 x 768 px Display with 16 bit color
- 3 Button Mouse
- Open GL Graphics Card with 16 MB Ram

##### Optimal specifications for Hardware

- 2 Ghz dual CPU
- 2 GB Ram
- 1920 x 1200 px Display with 24 bit color
- 3 Button Mouse
- Open GL Graphics Card with 128 or 256 MB Ram



### **3.2.3 Appearance Requirements**

As our system must be integrated in Blender, it shall comply with Blender standards. Indeed the environment places constraints on the product such as colours and visual components used and interface widget way of using.

### **3.2.4 Usability and Performance Requirements**

#### **Usability requirements**

- Ease of learning: The system must be easy to learn for both novices and experienced users.
- Efficiency of use: The system must be efficient for animators; this can be checked with the gain of time doing steps manually or with the system.
- Ease of remembering: Even a casual user is expected to remember about using the product.
- Overall satisfaction in using the product: Any user must feel satisfied with the system.
- Feedback: The user won't need too much feedback to feel confident that the product is actually accurately doing what he expects.

#### **Speed and latency requirements**

The response shall be fast enough to avoid interrupting the user's flow of thought. Any interface between the user and the automated system shall have a maximum response time of 2 seconds.

#### **Precision or Accuracy Requirements**

Phoneme should fit precisely to the text loaded by the user. The animation accuracy will depends on how many shapes will be created by the user.

### 3.2.5 Release and legal Requirements

#### Release Requirements

For our system to be integrated in Blender, we need to check carefully the coding structure, add comments and so on so it will be accepted by the Blender Community. We also have some constraints directly due to the software in which we want to integrate the system:

- All platforms operating

Description:	The product shall operate using all platforms.
Rationale:	Blender runs on all platforms and does not wish to change.
Fit criterion:	The product shall be approved as Blender compliant by the Blender Community.

- Open Source

Description:	The product shall be open source.
Rationale:	Blender is open source.
Fit criterion:	The product shall be approved as Blender compliant by the Blender Community.

We need to check the license of hypothetical reused components; it should be free, open source and all platforms.

#### Products That Can Be Reused

List of products or parts of products that can be legally copied or easily modified to be used in the project:

Software name	Company	Description
Papagayo	Lost Marble	Free open source software for 2D lip sync
Yolo	SourceForge.net	Free open source software for 2D lip sync
JLipSync	SourceForge.net	Free open source software for 2D lip sync
Sapi Lip Sync	Annosoft	Free open source console software doing speech recognition. Uses a Windows module to work. Doesn't work on non Windows platforms.

### **3.3 Context of use**

For more knowledge about our eventual users we made several surveys throughout forum post, online chats and so on, and it helped a lot for having a good idea of our target. And precisely in which conditions they may use the tool we are developing.

#### **3.3.1 Target Audience**

As we identified earlier the eventual user can be defined as:

- The beginner animators: They don't have much knowledge about 3D modelling and animating but which to experiment in an easy way the features that open source software like Blender can offer. This target audience needs user friendly panels, and gives up systematically when it's too complicate.
- The Intermediate animators: Self-taught persons or students, generally involved in a small scale project like a short clip either for pedagogical reasons or just for fun. Having the basic knowledge of the subject or seeking to acquire it, they are more likely disposed to further.
- The Expert animators: They know perfectly the subject and are aware of any improvement in the domain. They usually intend to reach realistic results and seek for great flexibility. As there are involved in big projects they must work in team, which means they must be able to share their settings, export, import them and so on.

#### **3.3.2 Locations and Time of usage**

For the beginners the location simply might be anywhere, as not involved in any production. Concerning the intermediates it's quite likely the same expect from the students who are usually located in universities or schools. And about the expert animators they tend to be in studios or offices. For all of our potential users the time of the using is related to those locations. The expert and students have timetable constraints, but for the beginner user it's not relevant.

#### **3.3.3 Reasons of usage**

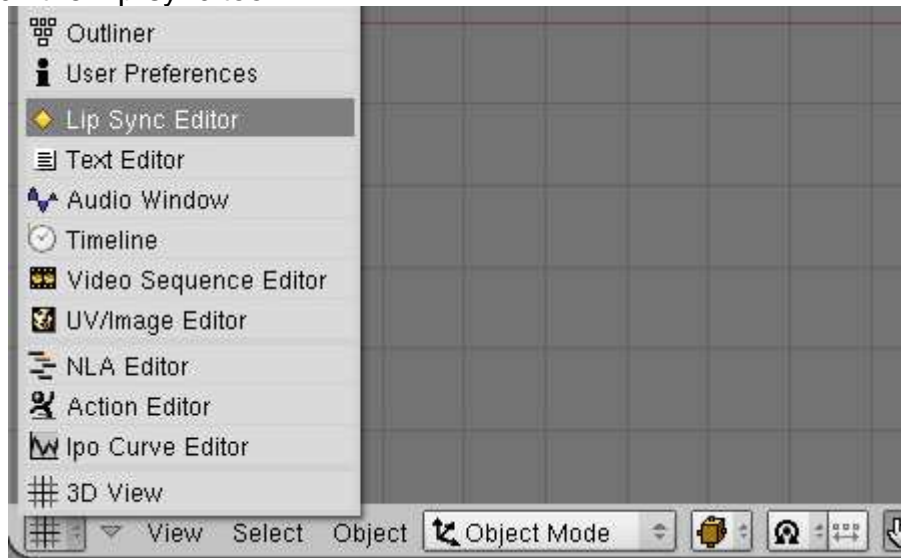
One of the most recurrent reasons that people gave us during our survey is the gain of time. Because traditionally when the user perform any lip-synching, accordingly to the timescale it may take a huge amount of time before reaching a good result. And that gain of time might be used for the rendering or any other process and so increase the quality of the production. At a lower level for the beginners and intermediate animators our tool allow them to make more advanced scenes as the lip-synching wouldn't be no more a bound that prevent them from animating such kind of scenes. For the "cartoonish " animations for example the accuracy doesn't need to be high as the realistic ones, in that way the default settings of our tool would be sufficient.

### 3.3.4 Usability

To use our tool the user should perform a few steps.

Storyboard:

1. Launch the Lip-sync tool:



**Illustration 5: Launching of the Lip-Sync Editor**

2. Add new Lip Sync file or load a new one



**Illustration 6: Adding new Lip-Sync project**

3. Add new Mapping file or load a new one



**Illustration 7: Adding new Mapping**

4. Load the audio file by browsing it on the hard drive disc



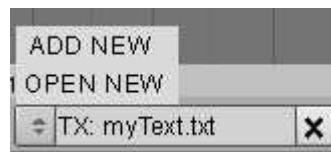
**Illustration 8: Audio list box**

The Lip-sync Editor is updated with a representation of the audio file along the timeline.



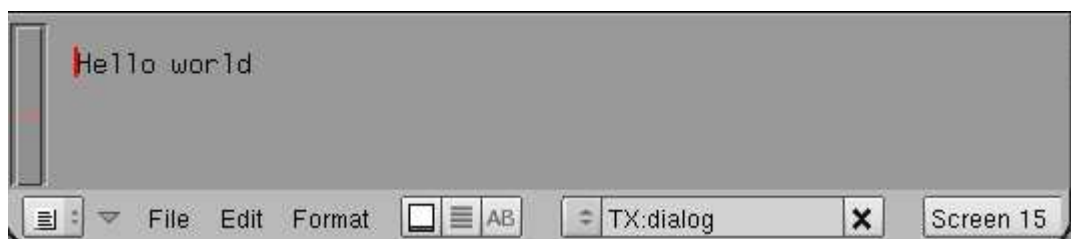
**Illustration 9: Audio strip**

5. Load the text file by browsing it on the hard drive disc, or create a new one



**Illustration 10: Text list box**

The Text Editor is filled with the loaded text, or is empty if a new one has been created; waiting for the user's input.



**Illustration 11: Text Editor**

6. Display on the Lip-Sync Editor the Phonemes and the words of the processed text and audio.



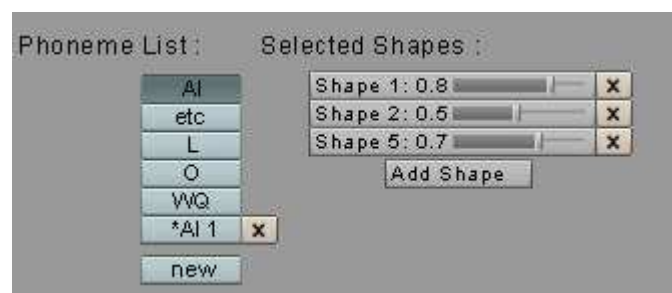
**Illustration 12: Lip-Sync Editor**

7. Display of the detected phonemes of the text and the customized one.



**Illustration 13: Phoneme list**

8. Selection of the Shape Keys to combine for each wanted Phoneme.



**Illustration 14: Mapping editor**

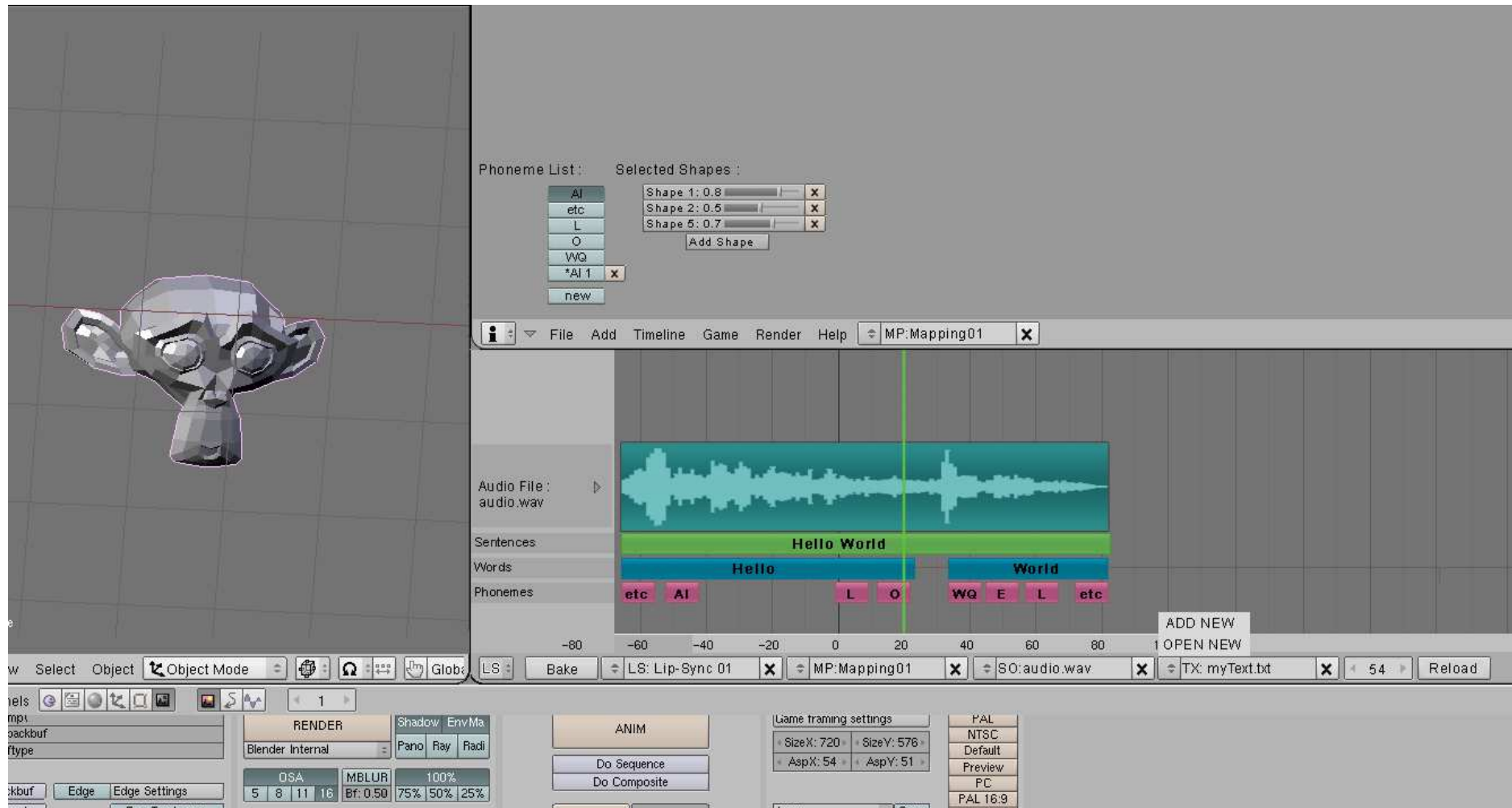
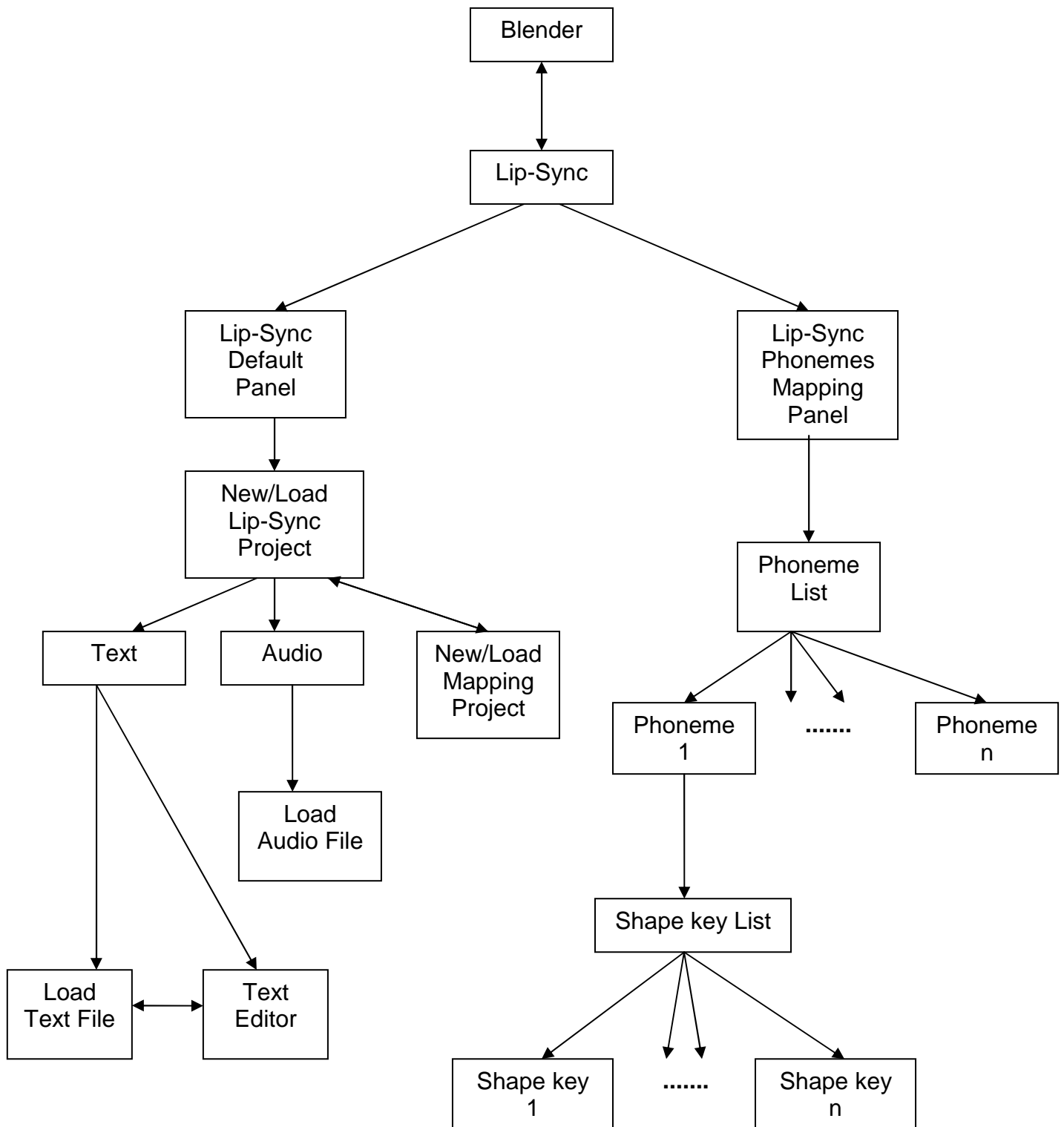


Illustration 15 : Lip-Sync Editor global view



**Illustration 16: Navigation Map**



## IV. Acceptance Tests

### 4.1 User Requirements

In order to check if user requirements have been met, we need to proceed to some tests called acceptance tests. Each test allows us to check an expected result from the system. For each test, the customer will be asked to follow a method and compare the actual result with the expected result in order to decide of a boolean outcome for this test: pass or fail.

#### 4.1.1 Functional tests

Test 1 : Create a lip sync project	
<i>Corresponding Requirement</i>	Requirement number : 1
<i>Purpose</i>	Test that the program can create a lip sync project.
<i>Method/Input</i>	Blender is launched; Lip sync editor is opened. The user chooses "add new" in the lip sync project list box.
<i>Expected results</i>	A lip sync project is created with a default name. The user is able to rename it by typing a name. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Abandon tests, noting the problem. Indeed, this step is crucial to follow the process and we have to fix it in order to deliver our software.

Test 2 : Select a lip sync project	
<i>Corresponding Requirement</i>	Requirement number : 2
<i>Purpose</i>	Test that the program can load an existing lip sync project.
<i>Method/Input</i>	Blender is launched; Lip sync editor is opened. The user selects an already existing lip sync project in the list box.
<i>Expected results</i>	The selected lip sync project is opened in the Lip sync editor. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Abandon tests, noting the problem. Indeed, this step is crucial to follow the process and we have to fix it in order to deliver our software.

Test 3 : Close a lip sync project	
<i>Corresponding Requirement</i>	Requirement number : 3
<i>Purpose</i>	Test that the program can close a lip sync project.
<i>Method/Input</i>	Blender is launched; Lip Sync editor is opened and a lip sync project is selected. The user clicks on the cross to the right of the lip sync list box.
<i>Expected results</i>	The selected lip sync project disappears from the list box. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem. The problem will have to be fixed but it will not prevent the user from performing the other tests.

Test 4 : Load sound file	
<i>Corresponding Requirement</i>	Requirement number : 4
<i>Purpose</i>	Test that the program can load a sound file.
<i>Method/Input</i>	Blender is launched; Lip sync editor is opened. The user chooses "open new" in the sound list box to browse his hard drive and select a .wav file, or selects an already loaded one in the list box.
<i>Expected results</i>	A window asks the user to select a .wav file located in his computer if the file hasn't been already loaded; then, this file is represented along the timeline in the Lip sync editor. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem and keeping in mind that, if the problem is not solved, text synchronization with audio will be compromised. Indeed, it is a crucial part of our application and will need a sound file to be loaded in order to work properly. The problem will have to be fixed in order to deliver our software.

Test 5 : Close sound file	
<i>Corresponding Requirement</i>	Requirement number : 5
<i>Purpose</i>	Test that the program can close a sound file.
<i>Method/Input</i>	Blender is launched; Lip Sync editor is opened and a sound is selected in the sound list box (a sound file is loaded in the lip sync editor). The user clicks on the cross to the right of the sound list box.
<i>Expected results</i>	The selected sound disappears from the sound list box. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem. The problem will have to be fixed but it will not prevent the user from performing the other tests.

Test 6 : Create a text file	
<i>Corresponding Requirement</i>	Requirement number : 6
<i>Purpose</i>	Test that the program can load a text file.
<i>Method/Input</i>	Blender is launched; Lip sync editor is opened. The user can select "Open new" in the text list box to browse his hard drive and select a .txt file, or can select the "Add new" control in the text list box and open the text editor to start using it.
<i>Expected results</i>	A window asks the user to select a .txt file located in his computer or the user can use Blender's text editor. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Abandon tests, noting the problem. Indeed, this step is crucial to follow the process and we have to fix it in order to deliver our software.

Test 7 : Load a text file	
<i>Corresponding Requirement</i>	Requirement number : 6
<i>Purpose</i>	Test that the program can load a text file.
<i>Method/Input</i>	Blender is launched; Lip sync editor is opened. The user selects a text in the text list box.
<i>Expected results</i>	Text is loaded in the "Lip Sync" editor: sentences, words and generated phonemes are represented along the timeline. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Abandon tests, noting the problem. Indeed, this step is crucial to follow the process and we have to fix it in order to deliver our software.

Test 8 : Close a text	
<i>Corresponding Requirement</i>	Requirement number : 7
<i>Purpose</i>	Test that the program can close a used text.
<i>Method/Input</i>	Blender is launched; Lip sync editor is opened and a text is selected in the text list box. The user clicks on the cross to the right of the text list box.
<i>Expected results</i>	The selected text disappears from the list box. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem. The problem will have to be fixed but it will not prevent the user from performing the other tests.

Test 9 : Adjust timing and text with the audio	
<i>Corresponding Requirement</i>	Requirement number : 8
<i>Purpose</i>	Test that the program allows user to adjust text with audio.
<i>Method/Input</i>	Blender is launched; Lip sync editor is opened. A lip sync project is selected in the lip sync project list box. Text and audio are loaded in the Lip sync editor.
<i>Expected results</i>	The user is able to move sentences and words on the timeline and to resize them according to the audio using mouse clicks and drag & drop operation. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem and keeping in mind that if the problem is not solved, animation will be compromised. Therefore, the problem will have to be fixed in order to deliver our software.

Test 10 : Reload text and audio	
<i>Corresponding Requirement</i>	Requirement number : 9
<i>Purpose/Input</i>	Test that the program can reload a text and a sound used.
<i>Method/Input</i>	Blender is launched; Lip sync editor is opened. A lip sync project is selected in the lip sync project list box. Text and audio are loaded in the Lip sync editor. The user clicks on reload.
<i>Expected results</i>	Text loaded in the "Lip Sync" editor is updated according to the new content of the selected text. Sound is updated according to the new content of the .wav file. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem. The problem will have to be fixed but it will not prevent the user from performing the other tests.

Test 11 : Set the starting frame of a lip sync project	
<i>Corresponding Requirement</i>	Requirement number : 10
<i>Purpose</i>	Test that the program can set the starting frame of a lip sync project
<i>Method/Input</i>	Blender is launched; Lip sync editor is opened. The user type the number of the starting frame he has decided in the "Set starting frame" text area.
<i>Expected results</i>	The starting frame of the lip sync animation is set to the frame chosen by the user. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem and keeping in mind that if the problem is not solved, the user will have to start his lip sync animation at the beginning of the scene. Therefore, the problem will have to be fixed in order to deliver our software.

Test 12 : Create a mapping project	
<i>Corresponding Requirement</i>	Requirement number : 11
<i>Purpose</i>	Test that the program can create a mapping project.
<i>Method/Input</i>	Blender is launched; Mapping editor is opened. The user chooses "add new" in the mapping project list box.
<i>Expected results</i>	A mapping project is created with a default name. The user is able to rename it by typing a name. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Abandon tests, noting the problem. Indeed, this step is crucial to follow the process and we have to fix it in order to deliver our software.

Test 13 : Select a mapping project	
<i>Corresponding Requirement</i>	Requirement number : 12
<i>Purpose</i>	Test that the program can load an existing mapping project.
<i>Method/Input</i>	Blender is launched; Mapping editor is opened. The user selects an already existing mapping project in the list box.
<i>Expected results</i>	The selected mapping project is opened in the Mapping editor. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Abandon tests, noting the problem. Indeed, this step is crucial to follow the process and we have to fix it in order to deliver our software.

Test 14 : Close a mapping project	
<i>Corresponding Requirement</i>	Requirement number : 13
<i>Purpose</i>	Test that the program can close a mapping project.
<i>Method/Input</i>	Blender is launched; Mapping editor is opened and a mapping project is selected. The user clicks on the cross to the right of the mapping list box.
<i>Expected results</i>	The selected mapping project disappears from the list box.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem. The problem will have to be fixed but it will not prevent the user from performing the other tests.

Test 15 : Link a lip sync project to a mapping project	
<i>Corresponding Requirement</i>	Requirement number : 14
<i>Purpose</i>	Test that the program can link a lip sync project to a mapping project.
<i>Method/Input</i>	Blender is launched, Lip sync editor is opened and a lip sync project is selected. The user selects a Mapping project in a list box.
<i>Expected results</i>	The chosen Mapping project is linked to the selected Lip sync project.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Abandon tests, noting the problem. Indeed, this step is crucial to follow the process and we have to fix it in order to deliver our software.



Test 16 : Create a new phoneme	
<i>Corresponding Requirement</i>	Requirement number : 15
<i>Purpose</i>	Test that the program can create a new phoneme.
<i>Method/Input</i>	Blender is launched; Mapping editor is opened. The user chooses "add new" in the phoneme list box.
<i>Expected results</i>	A phoneme is created with a default name. The user is able to rename it by typing a name. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem. The problem will have to be fixed but it will not prevent the user from performing the other tests.

Test 17 : Select a phoneme	
<i>Corresponding Requirement</i>	Requirement number : 16
<i>Purpose</i>	Test that the program can load an existing phoneme.
<i>Method/Input</i>	Blender is launched; Mapping editor is opened. The user selects an existing phoneme in the phoneme list box.
<i>Expected results</i>	The user is able to link the selected phoneme to a percentage of one or more shape key. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Abandon tests, noting the problem. Indeed, this step is crucial to follow the process and we have to fix it in order to deliver our software.

Test 18 : Delete a phoneme	
<i>Corresponding Requirement</i>	Requirement number : 17
<i>Purpose</i>	Test that the program can delete a phoneme.
<i>Method/Input</i>	Blender is launched; Mapping editor is opened and a phoneme is selected in the phoneme list box. The user clicks on the cross to the right of the phoneme list box.
<i>Expected results</i>	The selected phoneme disappears from the list box.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem. The problem will have to be fixed but it will not prevent the user from performing the other tests.

Test 19 : Link phonemes to several shape keys	
<i>Corresponding Requirement</i>	Requirement number : 18
<i>Purpose</i>	Test that the program can link each phonemes generated to one or more shape keys using percentage of them.
<i>Method/Input</i>	Blender is launched; Mapping editor is opened. The user has created one or more shape keys for an object. A phoneme is selected in the phoneme list box. The user selects a percentage of each shape keys he wants using scroll bars.
<i>Expected results</i>	The selected phoneme is linked to each shape key according to the percentage user has selected. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Abandon tests, noting the problem. Indeed, this step is crucial to follow the process and we have to fix it in order to deliver our software.

Test 20 : Bake animation	
<i>Corresponding Requirement</i>	Requirement number : 19
<i>Purpose</i>	Test that the program can “bake” an animation of an object according to the selected lip sync project.
<i>Method/Input</i>	Blender is launched; Lip sync editor and Mapping editor are opened. Text and sound are loaded in the Lip sync editor and have been synchronized. The user has created one or more shape keys for an object and at least one phoneme is linked to one shape key. The user clicks on the “bake” button in the Mapping editor.
<i>Expected results</i>	Animation is baked according to text and sound timing adjustments and percentage of shape keys selected for each phoneme. Key frames are created by the program. The user is able to proceed to other tests.
Actual Result (Pass/ Fail)	
Comments	
Actions to be undertaken if test fails	Complete the following acceptance tests, noting the problem and keeping in mind that, if the problem is not solved animation will be compromised. Therefore, the problem will have to be fixed in order to deliver our software.

#### 4.1.2 Appearance tests

As our system must be integrated in Blender, it shall comply with Blender standards.

Does our program respect Blender standards?	Yes	No	Comments
Colours			
Visual components			
Way of using			

Result (Pass/Fail):	Comments :
---------------------	------------

### 4.1.3 Usability and Performance tests

#### Usability tests

- Ease of learning

Is our system easy to learn (after reading the user manual)?	Yes	No	Comments
Non Blender user			
Blender user			

Result (Pass/Fail):	Comments :
---------------------	------------

- Efficiency of use

Does our system produce a significant gain of time for doing lip synchronization?	Time elapsed doing steps manually	Time elapsed doing steps with the system.	Comments

Result (Pass/Fail):	Comments :
---------------------	------------

- Ease of remembering

Does a casual user remember about using the product?	Yes	No	Comments

Result (Pass/Fail):	Comments :

- Overall satisfaction in using the product

Is user feeling satisfied with the system?	Yes	No	Comments

Result (Pass/Fail):	Comments :

### Speed and latency test

Is any interface between the user and the automated system having a maximum response time of 2 seconds?	Yes	No	Comments

Result (Pass/Fail):	Comments :

#### 4.1.4 Release and legal test

##### Release tests

- All platforms operating

Is the program all platforms operating?	Yes	No	Comments
Does the program operate on Windows?			
Does our program operate on Linux?			
Does our program operate on Mac?			

Result (Pass/Fail):	Comments :
---------------------	------------



- Open Source

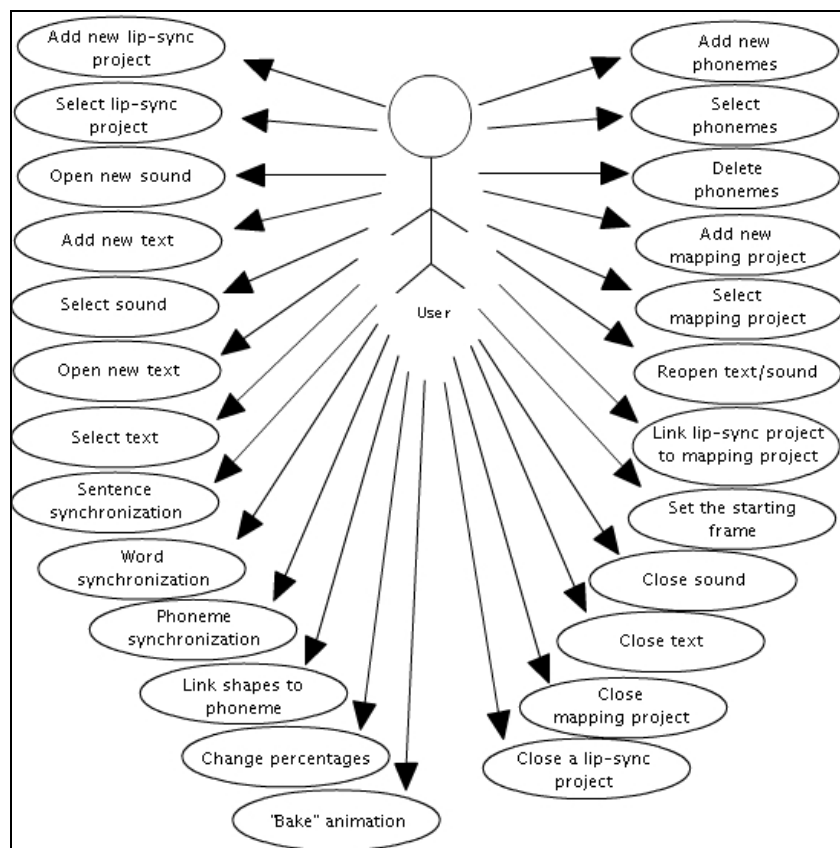
Does our program respects open source standards?	Yes	No	Comments

Result (Pass/Fail):	Comments :

## V. Design Description

The project has to be defined and has to contain breakdowns to develop in the best conditions. Consequently, we did several schemas by using UML methods. Therefore, firstly, we present a use-cases diagram describing each case with the pre and postconditions. Then, we present the class diagram with the description of each class. Finally, we made a sequence diagram of our project.

### 5.1 Use-Cases



Use-cases for our lip-sync interface

#### 1. Add new lip-sync project

PRECONDITIONS => Blender is launched and the lip-sync editor is opened.

POSTCONDITIONS => Create lip-sync project with no work in progress. This lip-sync project is selected.

#### 2. Select lip-sync project

PRECONDITIONS => Blender is launched and the lip-sync editor is opened.

At least one lip-sync project has been created.

POSTCONDITIONS => Select the chosen lip-sync project. The current lip-sync project becomes the selected project. Open an existing lip-sync project.

### 3. Open new sound

PRECONDITIONS => Blender is launched, the lip-sync editor is opened and the lip-sync project is selected.

POSTCONDITIONS => Open the chosen audio file.

Open an existing audio file.

### 4. Add new text

PRECONDITIONS => Blender is launched, the lip-sync editor is opened and the lip-sync project is selected.

POSTCONDITIONS => A new text file is created.

Create a new text file.

### 5. Select sound

PRECONDITIONS => Blender is launched, the lip-sync editor is opened and the lip-sync project is selected. At least one sound has been loaded

POSTCONDITIONS => Select the chosen sound. The current sound becomes the selected sound.

### 6. Open new text

PRECONDITIONS => Blender is launched, the lip-sync editor is opened and the lip-sync project is selected.

POSTCONDITIONS => Open the chosen text file.

Open an existing text file.

### 7. Select text

PRECONDITIONS => Blender is launched, the lip-sync editor is opened and the lip-sync project is selected.

At least one text has been created

POSTCONDITIONS => Select the chosen text. The current text becomes the selected text.

### 8. Sentence Synchronization

PRECONDITIONS => Blender is launched, the lip-sync editor is opened and the lip-sync project is selected. Text must be loaded.

POSTCONDITIONS => The sentences are moved.

Move the words and the phonemes in the same time.

### 9. Word Synchronization

PRECONDITIONS => Blender is launched, the lip-sync editor is opened and the lip-sync project is selected. Text must be loaded.

POSTCONDITIONS => The words are moved.

Move the phonemes in the same time.

#### 10. Phonemes synchronization

PRECONDITIONS => Blender is launched, the lip-sync editor is opened and the lip-sync project is selected. Text must be loaded.

POSTCONDITIONS => The phonemes are moved.

#### 11. Link shapes to a phoneme

PRECONDITIONS => Blender is launched, the mapping editor is opened and the mapping project is selected. Shape(s) must be created.

POSTCONDITIONS => A phoneme is linked with one or several shapes.

#### 12. Change percentages

PRECONDITIONS => Blender is launched, the mapping editor is opened and the mapping project is selected. Shape(s) must be created.

POSTCONDITIONS => The modification can be seen in the view port.

#### 13. "Bake" animation

PRECONDITIONS => Blender is launched, lip-sync and mapping editor are opened and a lip-sync and a mapping project are selected. Shapes must be linked with phonemes

POSTCONDITIONS => Add key-frames for each frame of the lip-sync animation and the animation is visible in the view port.

Impossible to undo this action.

#### 14. Add new phonemes

PRECONDITIONS => Blender is launched, the mapping editor is opened and the mapping project is selected.

POSTCONDITIONS => Add a new phonemes in the list of phonemes.

#### 15. Select phonemes

PRECONDITIONS => Blender is launched, the mapping editor is opened and the mapping project is selected. At least one phoneme has been created.

POSTCONDITIONS => Select the chosen phoneme. The current phoneme becomes the selected phonemes.

#### 16. Delete phonemes

PRECONDITIONS => Blender is launched, the mapping editor is opened and the mapping project is selected.

POSTCONDITIONS => Delete the selected phoneme from the list of phonemes.

#### 17. Add new mapping project

PRECONDITIONS => Blender is launched and the mapping editor is opened.

POSTCONDITIONS => Create a new mapping project with no work in progress. This mapping project is selected.

#### 18. Select mapping project

PRECONDITIONS => Blender is launched and the mapping editor is opened. At least one mapping project has been created.

POSTCONDITIONS => Select the chosen mapping project. The current mapping project becomes the selected mapping project.

The mapping project is selected.

#### 19. Reopen text/sound

PRECONDITIONS => Blender is launched, the lip-sync editor is opened and a lip-sync project is selected. A text or a sound must be opened before reloading.

POSTCONDITIONS => The text and/or the sound are reloaded.

The sentences and the words change if the text is reloaded. The sound graph change is the sound is reloaded.

#### 20. Link a lip-sync project to a mapping project

PRECONDITIONS => Blender is launched, lip-sync and mapping editor are opened and a lip-sync and a mapping project are selected. A lip-sync project and a mapping project have to exist before linking them.

POSTCONDITIONS => The lip-sync project and the mapping project are linked.

#### 21. Set the starting frame

PRECONDITIONS => Blender is launched and lip-sync editor is opened and a lip-sync project is selected.

POSTCONDITIONS => Change the beginning of the animation from the lip-sync project.

#### 22. Close sound

PRECONDITIONS => Blender is launched and lip-sync editor is opened and a lip-sync project is selected.

The sound is opened.

POSTCONDITIONS => Close the current sound.

#### 23. Close text

PRECONDITIONS => Blender is launched and lip-sync editor is opened and a lip-sync project is selected.

The text is opened.

POSTCONDITIONS => Close the current text.

#### 24. Close mapping project

PRECONDITIONS => Blender is launched and mapping editor is opened and a mapping project is selected.

The mapping project is opened.

POSTCONDITIONS => Close the current mapping project.

#### 25. Close a lip-sync project

PRECONDITIONS => Blender is launched and lip-sync editor is opened and a lip-sync project is selected.

The lip-sync project is opened.

POSTCONDITIONS => Close the lip-sync interface.

### 5.2 UML Class chart:

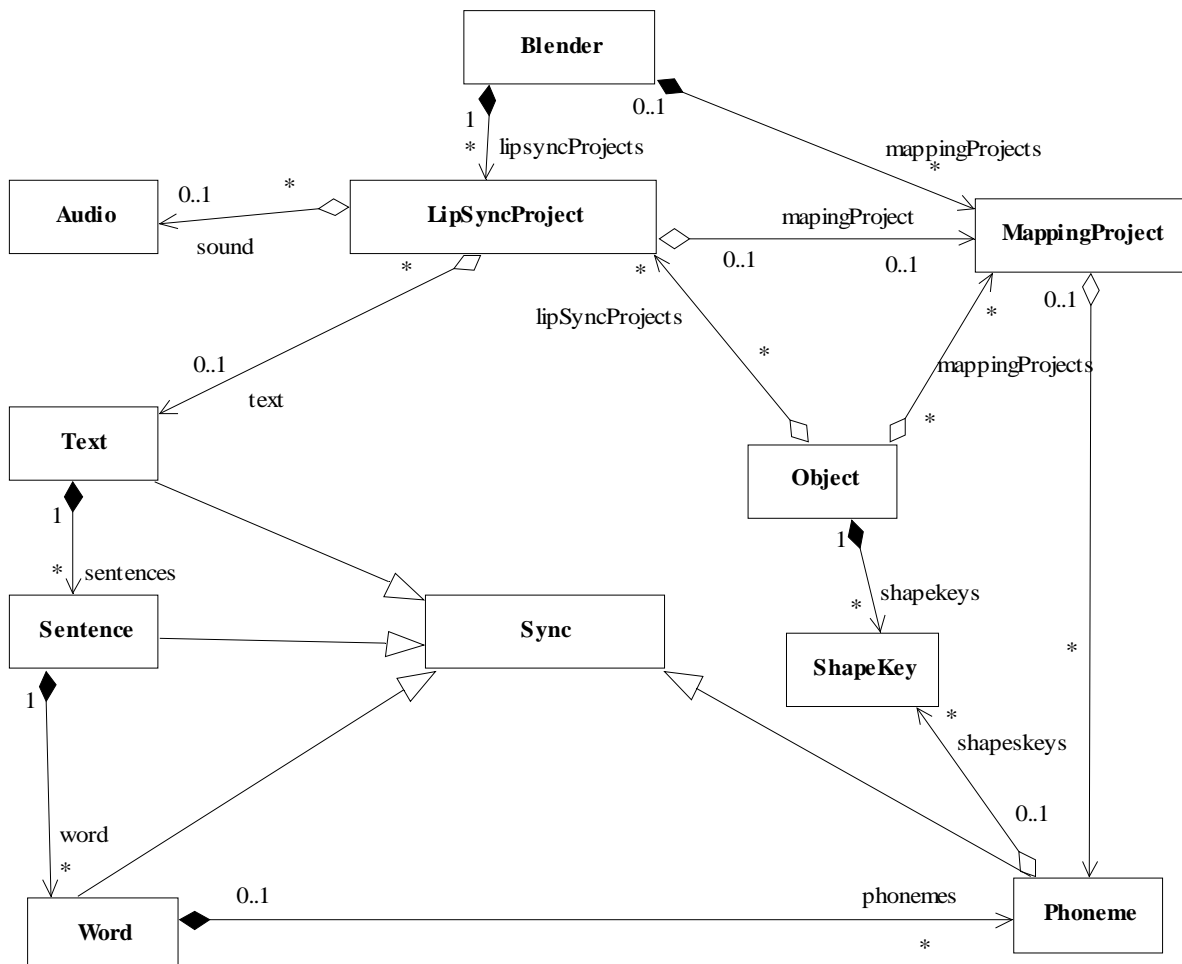


Illustration 17: Class Diagram

### Comment:

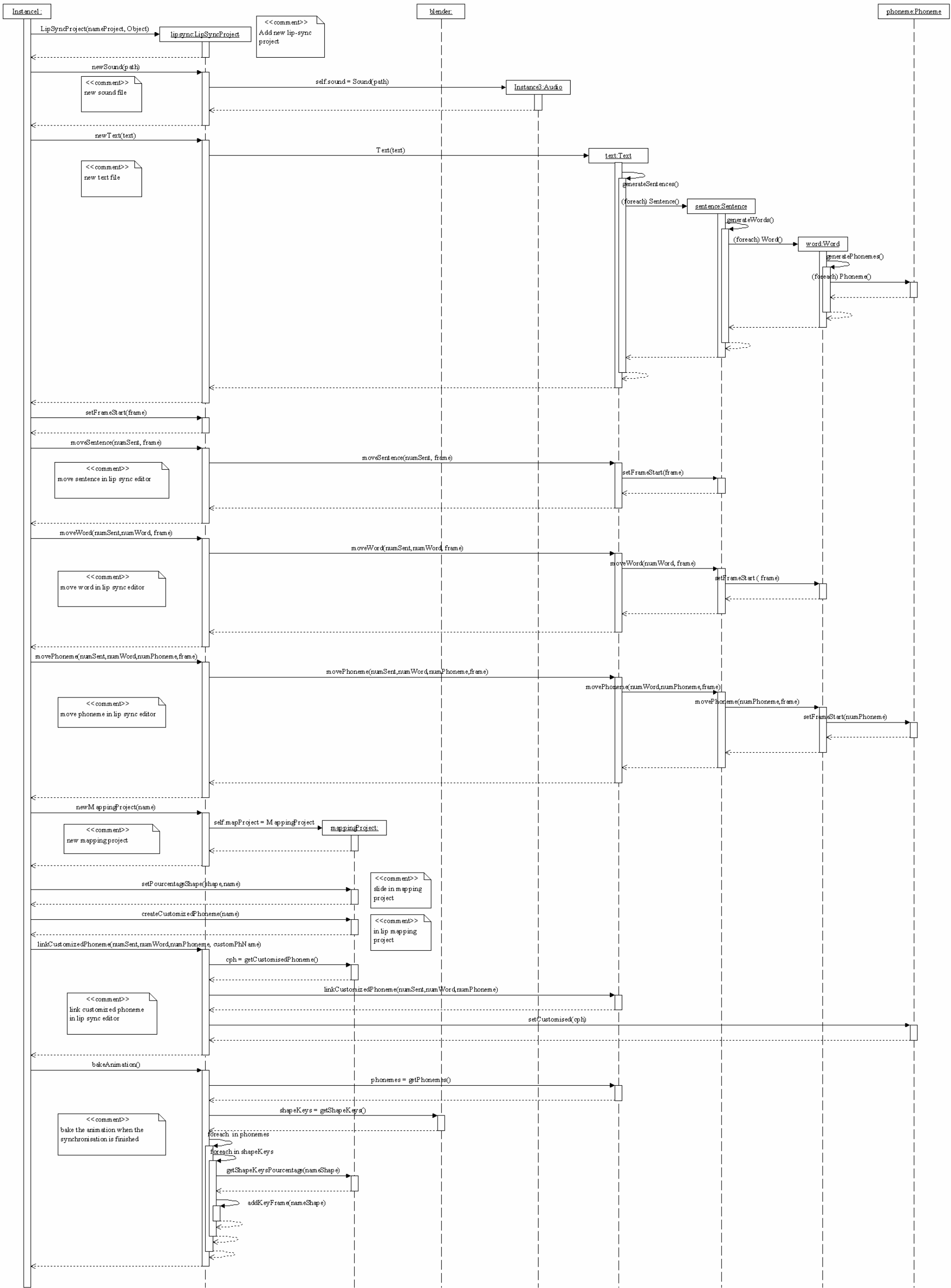
*Blender* and *Object* and *ShapeKey* and *Audio* are class already created. We will re-use the code.

### Description Class:

- *Blender* : this is the main class which contains all instances.
- *Object* : this is an object on which we can apply shape keys, that means *Lattices* and meshes.
- *ShapeKey* : this is a state of the *Lattices* or *Meshes*.
- *Audio* : this is an audio data used by the lip-sync project.
- *Sync* : this is a super class of *Text*, *Sentence*, and *Word*. The common attributes and methods are used to synchronize with the audio file.
- *Text* : this is a group of sentence instances.
- *Sentence* : this is a group of word instances.
- *Word* : this is a group of phoneme instances.
- *MappingProject* : this is an object to manage the shape keys combination mapped to phonemes.
- *LipSyncProject* : this is the main class of our project. It's to synchronize the text with the audio.

### 5.3 UML sequence

This following sequence diagram represents the sequences of the main use-cases of the project.





## 5.4 Implementation

Blender isn't Object Oriented conception so we should adapt UML conception. We re going to explain how to integrate our project in blender source code and how to separate process to GUI (Graphical User Interface).

The blender architecture is described on blender.org in this way: **“It has a strictly organized 'Data Oriented' structure, almost like a database, but with some Object Oriented aspects in it. It was entirely written in plain C”**. The persistence is managed by module called makeSDNA. We should add data block for each class defined in our class chart. The data block is a C *struct* type with multiples of 8 bytes in size. We should add class properties in its data block so it could be saved.

The class methods will be written with C *functions*. For each one, the first argument should be a pointer to data block. That way, we can do our development in Object oriented style: the first argument is like “this” or “self” in object programming. Furthermore Instead of inheritance mechanism, we will use compositions.

### Code organization

The code will be separated in three sections: data and GUI and edit. Their interaction is shown bellow.

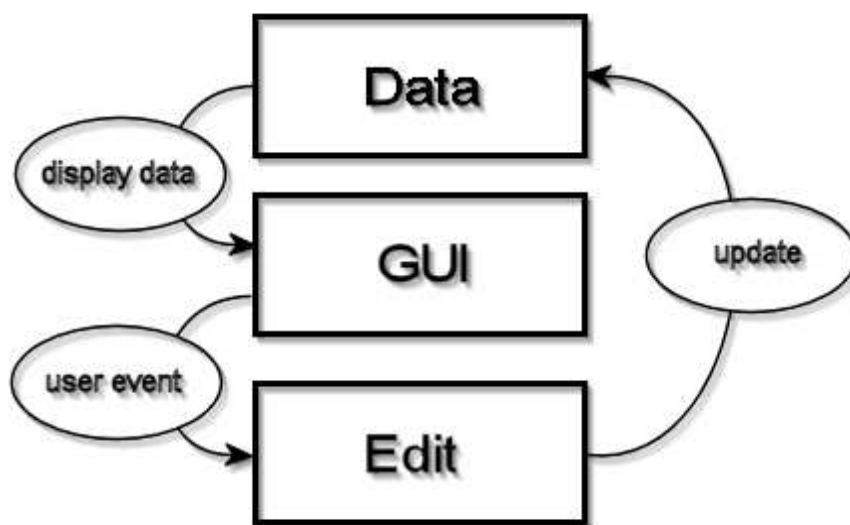


Illustration 18: Sections interaction

The data section will be in **DNA\_lipsynctypes.c**. In this file, we regroup the data blocks for each class. And the GUI definition should be in **drawlipsync.c**. And finally, in **editlipsync.c** we add class methods. As we will want to integrate the official source, we have to match blender code design.

## *VI. User Manual*

### Table of content

<i>VI. USER MANUAL</i> .....	<b>50</b>
6.1 Introduction: Welcome .....	<b>50</b>
6.2 Open the Lip-Sync Editor .....	<b>51</b>
6.3 Create/Select a lip-sync project .....	<b>52</b>
6.4 Load an audio file .....	<b>52</b>
6.5 Create/Load a text file .....	<b>52</b>
6.6 Adjust timing by moving sentences, words and phonemes .....	<b>53</b>
6.7 Reload text and sound used .....	<b>53</b>
6.8 Set the starting frame of your lip-sync project .....	<b>54</b>
6.9 Create/Select a mapping project .....	<b>54</b>
6.10 Link the phonemes to one or several shape keys .....	<b>55</b>
6.11 Create a new phoneme and delete a created phoneme .....	<b>55</b>
6.12 Bake animation .....	<b>56</b>
6.13 Your animation is ready .....	<b>56</b>

## **First contact with the Lip-Sync Editor in Blender**

### *6.1 Introduction: Welcome*

Welcome in the Lip-Sync Editor in Blender 3D.

- *What is the Lip-Sync Editor?*

This editor is an interface helping you to create a lip-synchronization in a scene for your 3D model. Indeed, you will be able to use a voice and synchronize it with a text of your choice in order to apply it to your 3D character. This makes easy to create realistic dialogs.

- *Why a Lip-Sync Editor in Blender 3D?*

Lip-synching is hard and takes time to produce. Therefore, its implementation in a free open-source 3D software will allow users from all platforms to experiment dialog animations in a easy way while saving time.

- *Who is the target?*

The user is a 3D modelling software user. She or he can be either a novice, or a user with intermediate level or with high skills in 3D animation. Indeed, our new editor will allow a beginner to learn the animation basics with a simple mesh. An intermediate user will be able to improve her or his animation overall quality. And an expert animator will enjoy the flexibility and the gain of time resulting from the use of the Lip-Sync Editor for her or his work.

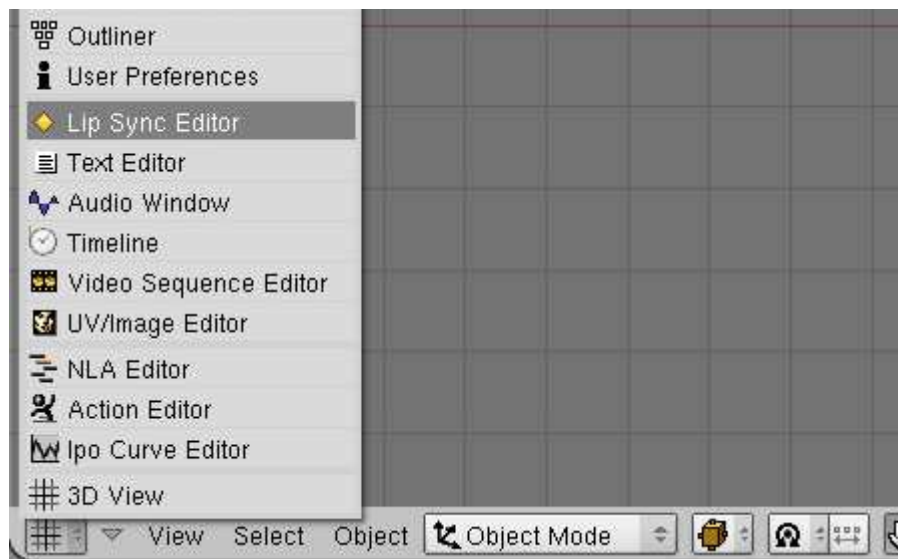
The aim of this feature is to fit as well non-professional requirements as professional. Although it is recommended to have skills in Blender, it is not compulsory as long as you have read the Lip-Sync Editor user manual.

Thanks to this editor, you will be able to create animation of lips of your character easily and fast.

This guide will help you to know how to use the Lip-Sync Editor explaining each step in details to do the lip-sync process with no difficulty.

## 6.2 Open the Lip-Sync Editor

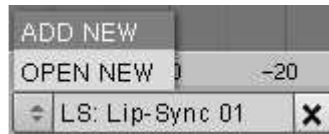
After opening Blender, click on the list of the editor and select *Lip-Sync Editor* (Illustration 19 : Launch Lip-Sync Editor).



**Illustration 19 : Launch Lip-Sync Editor**

### 6.3 Create/Select a lip-sync project

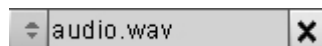
- After opening the editor, you have to open a new lip-sync project. To do that, click in the lip-sync project in the list box (Illustration 20: Lip-Sync project list box). Then you have the choice either you select an existing lip-sync project by clicking on a project in the list box, or you create a new lip-sync project by clicking on ADD NEW.



**Illustration 20: Lip-Sync project list box**

### 6.4 Load an audio file

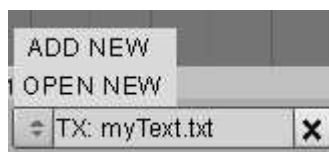
- Then, you have to load an audio file relating with the animation. Click on the audio list box (Illustration 21: Audio list box). Either, you select an existing file in the list loaded in Blender, or you select a new file that is not in the list by choosing a file in the browse window. Then, an audio graph is displayed in the editor



**Illustration 21: Audio list box**

### 6.5 Create/Load a text file

- You also have to load texts for your lip-sync project. Click on the text file list box (Illustration 22: Text list box). Then, you have also two solutions. Either you can create a new text file in Blender by selecting ADD NEW in the list box, or you can select OPEN NEW to open a text file from your hard drive. Then, the text is splitted in three parts: sentences, words and phonemes. These parts are displayed in different colours on the lip-sync editor (Illustration 23: Splitted text on Lip-Sync Editor).



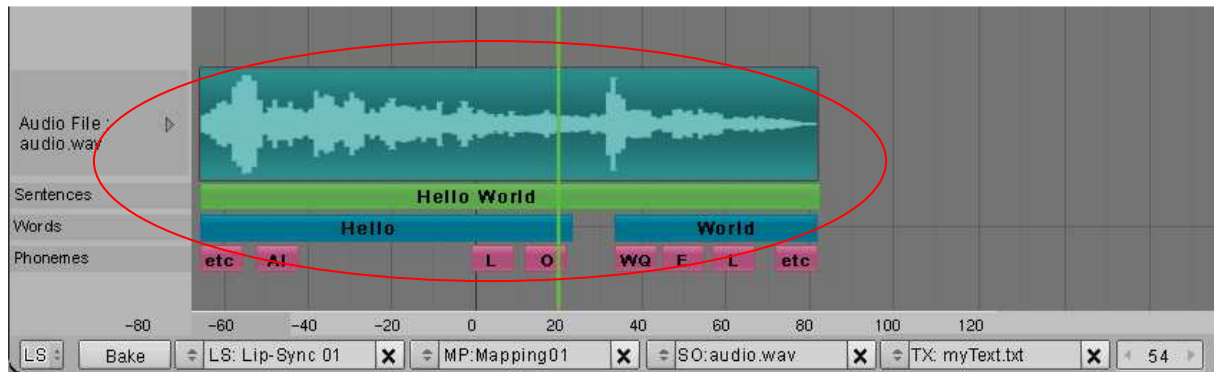
**Illustration 22: Text list box**



**Illustration 23: Splitted text on Lip-Sync Editor**

## 6.6 Adjust timing by moving sentences, words and phonemes

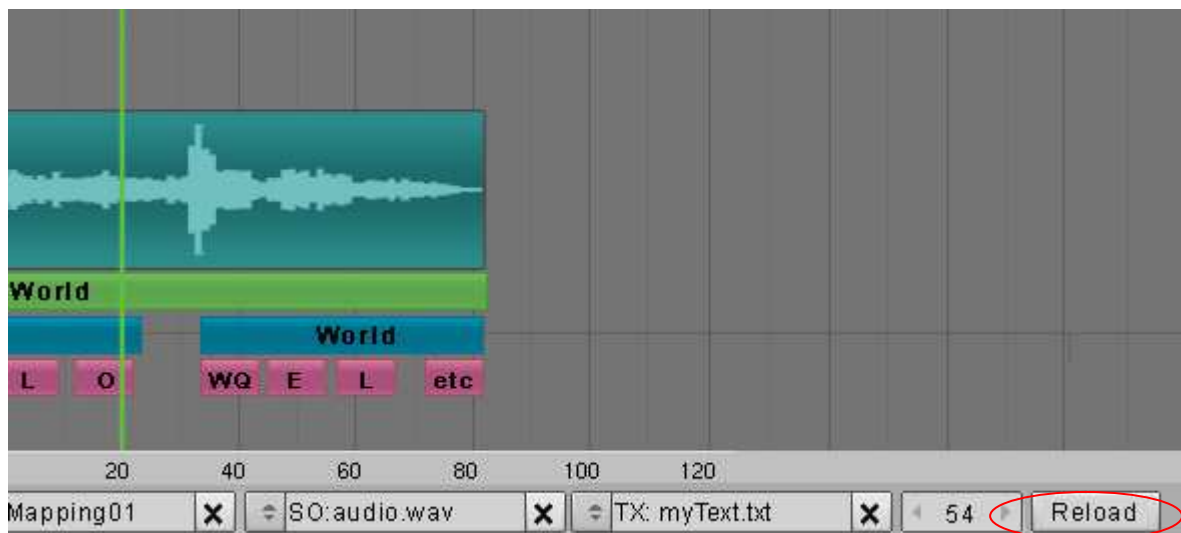
- Now, all the elements that you need to do the lip-sync are loaded and you can begin to work. The next step is to adjust the timing between the animation and the sound. To do that, click on a sentence, a word, or a phoneme to move them (Illustration 24: Adjusting in Lip Sync Editor). If you move a sentence, the word of this sentence and the phonemes of these words move in the same time. In addition, if you move a word, you move the phonemes of this word in the same time.



**Illustration 24: Adjusting in Lip Sync Editor**

## 6.7 Reload text and sound used

- If you have modified your audio file and/or your text file, you have to update it/them in Blender. To do that, click on the RELOAD BUTTON (Illustration 25: Reloading text and sound). You will see that your files have been updated: The elements will be modified if the audio file and/or the text file have been updated.



**Illustration 25: Reloading text and sound**

### 6.8 Set the starting frame of your lip-sync project

The next step is to enter the number of the first frame for your lip-sync project. (Illustration 26: Set starting frame) 0 is the number enter by default when you create a new lip-sync project. This number means the lip-sync animation will begin at the frame 0. If you want to edit it, click on this number and enter the number of your choice. After that, your lip-sync animation will begin at the frame you have chosen.

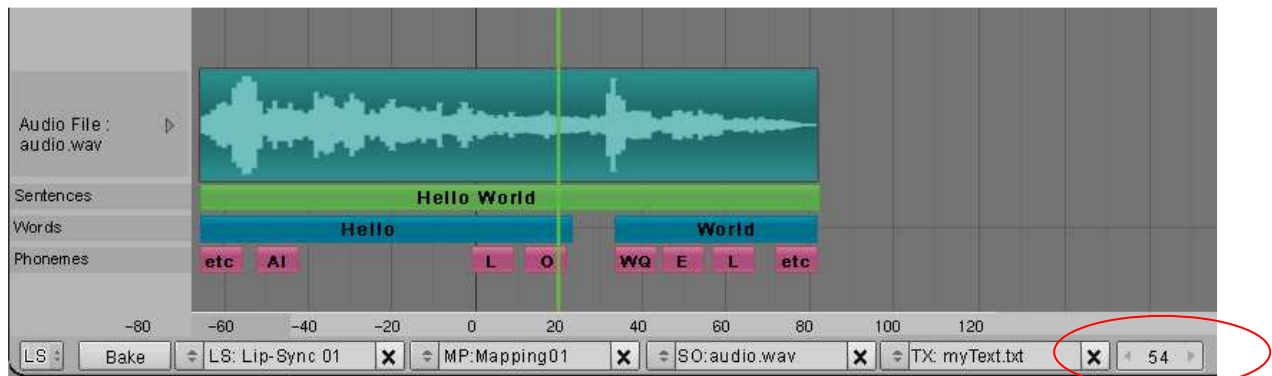


Illustration 26: Set starting frame

### 6.9 Create/Select a mapping project

Now, you have to create a mapping project by clicking in the list box for the mapping project (Illustration 27: Selecting a Mapping project) and clicking on ADD NEW. You can also select an existing mapping project by clicking on one of them. After creating or selecting a mapping project, the Mapping Editor is displayed with the current phonemes used (Illustration 28: Phoneme list). The Mapping editor allows you to link one or several “shapes keys” with a phoneme.



Illustration 27: Selecting a Mapping project

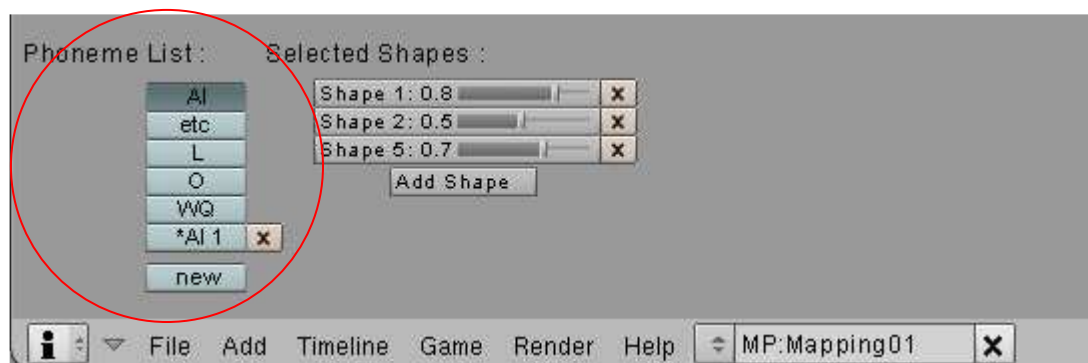
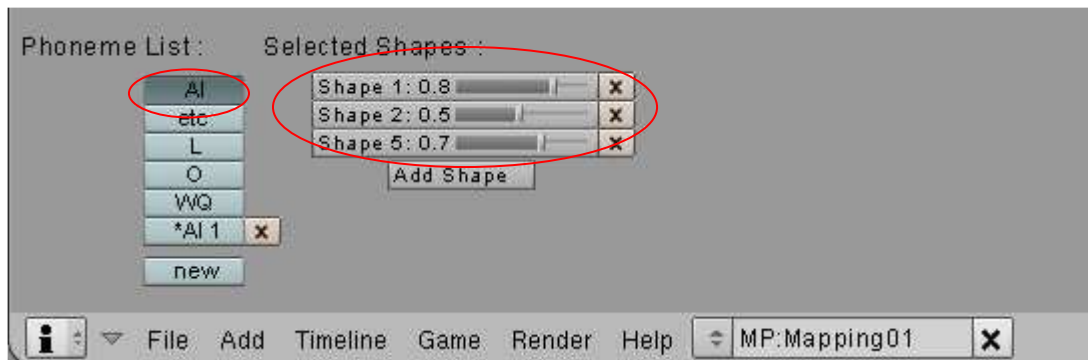


Illustration 28: Phoneme list

### 6.10 Link the phonemes to one or several shape keys

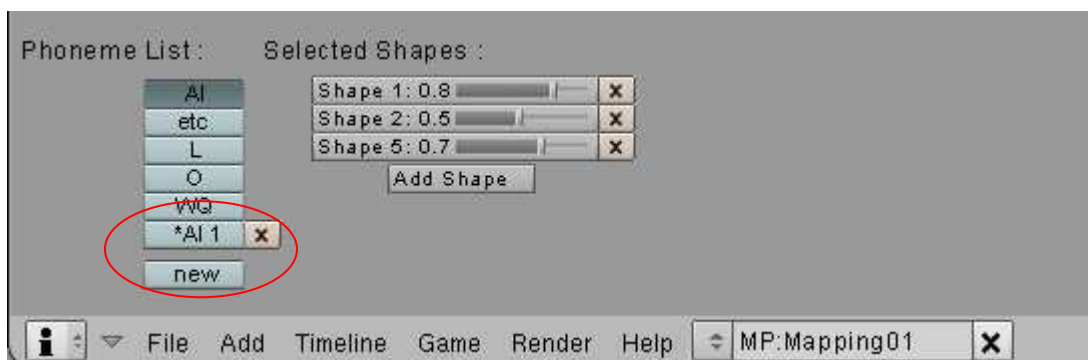
- Click on a phoneme. Then, use the scroll bar to change the percentage of the shape (Illustration 29: Phoneme settings). This will link this shape key with your selected phoneme.



**Illustration 29: Phoneme settings**

### 6.11 Create a new phoneme and delete a created phoneme

- You can also add your own phonemes and delete them. To create a new one, in the Mapping Editor, click ADD NEW on the phoneme list box. Then, a new phoneme appears with a default name. Of course, you can change the name of this new phoneme. You can also delete your own created phoneme. Click on your phoneme that you want to delete, then, click on the cross to the right of the phoneme list box. (Illustration 30: customized phonemes)



**Illustration 30: customized phonemes**



### 6.12 Bake animation

- Now, you are in the last step of the lip-sync process with the Lip-Sync Editor. To save time from your CPU and to save your lip-sync animation, you have to “bake” your animation. To do that, click on the BAKE button to bake the animation located in the Mapping Editor (Illustration 31: Baking animation). Then, each frame will contain a key frame.



**Illustration 31: Baking animation**

### 6.13 Your animation is ready

- Now, the lip-sync process is over and your lip-sync animation is ready to be exported. You can repeat all these steps for another character.



## *VII. Appendix*

### **7.1 Glossary**

**Keyframe:** Keyframe is a frame in an animated sequence of frames that was drawn or otherwise constructed directly by the user. When all frames were drawn by animators, the senior artist would draw these frames, leaving the "in between" frames to an apprentice. Now, the animator creates only the first and last frames of a simple sequence; the computer fills in the gap. This is called twining (From [wiki.blender.org](http://wiki.blender.org)).

**Lattice:** A lattice is a structure of points for carrying out free-form deformations on any deformable object. To create deformation effects, you edit the lattice by moving, rotating, or scaling the lattice structure, or by directly manipulating the lattice points. In general, you create effects by editing any of the lattice deformer's attributes (From Maya documentation).

**Mapping:** Mapping, in the context of our project, is the action of synchronize a phoneme with one or more shape keys according to a timing.

**Mesh:** It's a 3D object, which is a network of vertices, edges, and faces; it can be manipulated to transform a simple primitive shape into something very complex.

**Lip-sync:** **Lip-sync** or **Lip-synch** (short for **lip synchronization**) is a technical term for matching lip movements with voice. The term refers both to a technique often used during musical performances and the production of film and Television programs, and to the problem of maintaining video and audio signals synchronized during post-production and transmission. It is also used to match lip movements of animated characters (including computer facial animation) (From [www.wikipedia.org](http://www.wikipedia.org)).

**Phoneme:** In human language, a phoneme is the theoretical representation of a sound. It is a sound of a language as represented (or imagined) without reference to its position in a word or phrase. A phoneme, therefore, is the conception of a sound in the most neutral form possible (...) (From [www.wikipedia.org](http://www.wikipedia.org)).

**Shape keys:** Shape keys store different shapes of a base mesh. Other 3d applications may call these 'morph targets' or 'blend shapes'. In Blender, we have both relative shape keys and absolute shape keys (From [wiki.blender.org](http://wiki.blender.org)).

## 7.2 Table of illustrations

Illustration 1: General process .....	5
Illustration 2: Project Plan.....	8
Illustration 3: Event list of inputs and outputs .....	15
Illustration 4: Use-cases from the user's point of view .....	15
Illustration 5: Launching of the Lip-Sync Editor .....	20
Illustration 6: Adding new Lip-Sync project.....	20
Illustration 7: Adding new Mapping .....	20
Illustration 8: Audio list box .....	20
Illustration 9: Audio strip .....	21
Illustration 10: Text list box .....	21
Illustration 11: Text Editor.....	21
Illustration 12: Lip-Sync Editor .....	21
Illustration 13: Phoneme list .....	22
Illustration 14: Mapping editor .....	22
Illustration 15 : Lip-Sync Editor global view.....	23
Illustration 16: Navigation Map .....	24
Illustration 17: Class Diagram .....	46
Illustration 18: Sections interaction.....	49
Illustration 19 : Launch Lip-Sync Editor .....	51
Illustration 20: Lip-Sync project list box .....	52
Illustration 21: Audio list box .....	52
Illustration 22: Text list box .....	52
Illustration 23: Splitted text on Lip-Sync Editor .....	52
Illustration 24: Adjusting in Lip Sync Editor .....	53
Illustration 25: Reloading text and sound.....	53
Illustration 26: Set starting frame.....	54
Illustration 27: Selecting a Mapping project .....	54
Illustration 28: Phoneme list .....	54
Illustration 29: Phoneme settings .....	55
Illustration 30: customized phonemes.....	55
Illustration 31: Baking animation.....	56