MINIGAMES in XIMPEL

XIMPEL is an interactive video platform that enables users to create games with their own storylines. One of its features is the modularity, as XIMPEL can incorporate the user's own minigames written in Flex (Action Script). They can be added to the playlist in a similar manner as other media types. This gives endless possibilities of customizing the main application.

1. Creating a Minigame

Requires programming skills in Flex (Action Script and MXML)

1.1 Making minigames compatible with Ximpel Player

Depending on the version of Action Script we use, one of these pieces needs to be added to minigame's code:

[*ActionScript 2*]

```
var outgoing_lc:LocalConnection = new LocalConnection();
var status = "minigame_done";
var score:int;
minigame.sendEndGameMessage = function(Void):Void
{ outgoing_lc.send("minigame_status", "onMinigameComplete", status,score); }
```

[*ActionScript 3*]

```
import flash.net.LocalConnection;

private var outgoing_lc:LocalConnection = new LocalConnection();
private function sendEndGameMessage():void
{
private var status:String = "minigame_done";
private var score:int;
outgoing_lc.send("minigame_status", "onMinigameComplete", status, score);
}
```

Via a local connection with the name "*minigame_status*", the function *onMinigameComplete* is called with 2 arguments: *status* and *score*. If status "*minigame_done*" is sent, XIMPEL understands, that the minigame has been completed and it sends the score to the main application (this way general score can be updated with results from each minigame). The way the score is counted, should be indicated in minigame's code.

Once MXML file is ready, we need to create an SWF file, which can be easily done using Flex SDK.

1.2 How Does It Work?

One of XIMPEL's features is the possibility to register custom media types. To program a custom media type you need to create a class that implements the *IMediaType* interface. In order to use minigames in Ximpel Player, *Minigame* class has been created (see Appendix A). Every time a new minigame is added to the playlist, a new object is created with its local connection, that enables sending the status and score of each minigame to the main application.

To make this custom media type accessible in XimpelPlayer, it has been registered in the main application:
```
var minigame:Minigame = new Minigame();
minigame.addEventListener(MediaScoreEvent.SCORE_RESULT,updateMediaScore);
myXimpelPlayer.registerMediaType(minigame);
```

2. Templates

Does not require programming skills

A couple of minigame templates is available for users. Customizing them does not require any programming skills, so that everyone can easily create their own games. The templates use dynamic XML files, which include paths to the files and other variables. In folder named '*assets*', you can find directories for each type of a minigame. This is where the files are being stored.

2.1 Memory Game

Does not require programming skills

In *assets/memory* you can find *memorycards.xml*. Do not change the name of the file! An exemplary code looks as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<memoryCards>

<welcometxt>
<txt>Find all pairs. Less clicks = higher score!</txt>
</welcometxt>

<genericCardSide>
<url>mozart.jpg</url>
</genericCardSide>

<card>
<url>mozart1.jpg</url>
</card>
<card>
<url>mozart2.jpg</url>
</card>
<card>
<url>mozart3.jpg</url>
</card>

</memoryCards>
```

In order to create a memory game with your pictures, you need to change the names of the files within <url> tags. *genericCardSide* is the picture that will be displayed if the card is not flipped.
Within <welcometxt><txt> tag you can change text with instructions for your memory game.

Requires some programming skills

It is also possible to adjust memory game by changing the main code in *memory.mxml* file (see Appendix B). Such adjustments include:
- how the score is counted - in the original version, total clicks influence the score (less clicks = higher score); the formula used to count the score: *Math.floor((20 * cardPairsTaken) / totalClicks)*; this can be changed in the code;
- changing the size of images; in the original version they are displayed as 100x100px.

2.2 Matching Game

Does not require programming skills

In *assets/matching* you can find *matchingImages.xml*. Do not change the name of the file! An exemplary code looks as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<matchingImages>

<welcometxt>
<txt>Match events in Mozart's life with the correct date by dragging images from the top row onto matching dates. Less clicks = higher score!</txt>
</welcometxt>

<image>
<url>born.jpg</url>
<match>1756.jpg</match>
</image>

<image>
<url>died.jpg</url>
<match>1791.jpg</match>
</image>

</matchingImages>
```

In order to create a matching game with your pictures, you need to change the names of the files within <url> and <match> tags. Make sure, that the matching images are within one <image> tag!
Within <welcometxt><txt> tag you can change text with instructions for your matching game.

Requires some programming skills

It is also possible to adjust matching game by changing the main code in *matching.mxml* file (see Appendix C). Such adjustments include:
- how the score is counted - in the original version, total clicks influence the score (less clicks = higher score); the formula used to count the score: *Math.floor((8 * pairsTaken) / totalClicks)*; this can be changed in the code;
- changing the size of images; in the original version they are displayed as 100x100px. The semi-transparent image displayed when dragging is called a *drag proxy*. When changing the size of the image, you might also want to change the size of the proxy in *initiateDrag* function.

2.3 Sorting Game

Does not require programming skills

In *assets/sorting* you can find *sortingImages.xml*. Do not change the name of the file! An exemplary code looks as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<sortingImages>

<welcometxt>
<txt>Sort Mozart's pieces from the earliest to the latest. Less clicks=higher score!
</txt>
</welcometxt>

<image>
<url>bastien.jpg</url>
<priority>1</priority>
</image>
<image>
<url>thamos.jpg</url>
<priority>2</priority>
</image>
<image>
<url>dongiovanni.jpg</url>
<priority>3</priority>
</image>

</sortingImages>
```

In order to create a sorting game with your pictures, you need to change the names of the files within <url> tag. The order of the images does not matter, but it is important that you give a priority number to each of your pictures. The first picture to be sorted should have a priority of 1.
Within <welcometxt><txt> tag you can change text with instructions for your game.

Requires some programming skills

It is also possible to adjust matching game by changing the main code in *matching.mxml* file (see Appendix D). Such adjustments include:
- how the score is counted - in the original version, total clicks influence the score (less clicks = higher score); the formula used to count the score:
*Math.floor((8 * pairsTaken) / totalClicks);* this can be changed in the code;
- changing the size of images; in the original version they are displayed as 120x120px. The semi-transparent image displayed when dragging is called a *drag proxy*. When changing the size of the image, you might also want to change the size of the proxy in *initiateDrag* function.

2.4 Questionnaire

Does not require programming skills

One of the available templates is a questionnaire. It allows users to create a survey or a test with a random number of questions and answers.

In *assets/questionnaire* you can find *questions.xml*. Do not change the name of the file! An exemplary code looks as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<test>

<welcometxt>
<txt>Choose correct answer. When ready, click the submit button!</txt>
</welcometxt>

<question>
<q>When did Mozart live?</q>
<answer value="false">1754 - 1789</answer>
<answer value="false">1756 - 1789</answer>
<answer value="true">1756 - 1791</answer>
</question>

<question>
<q>What was the name of Mozart's wife?</q>
<answer value="true">Constanze</answer>
<answer value="false">Aloysia</answer>
</question>

</test>
```

In order to create your own questionnaire, you need to state your questions within <q> tag and all the possible answers within <answer> tags. Do not forget to add a value="true" or "false" for each answer!

Within <welcometxt><txt> tag you can change text with instructions for your questionnaire.

The full code of a questionnaire template is available in Appendix E.

3. Adding a Minigame to the Playlist

To add a minigame to the playlist, simply use <minigame file="sorting.swf"/> within <media> tags, adjusting the name of your swf file.

Requires some programming skills

It is possible to add to the playlist more than just one minigame of a kind and it can be done in few steps (the example is for a sorting game, but works the same for each minigame):

1) Create your first sorting game.
2) Find the file *sorting.swf* and rename it to *sorting1.swf*. From now on this is your first sorting game.
3) Open *sorting.mxml* file and find a line with url for a HTTP Service (one of the first lines): *url="assets/sorting/sortingImages.xml"* . Change the path to *assets/sorting/sortingImages2.xml*.
4) Using *make_sorting.bat*, create your new *sorting.swf* file. This step requires using Flex SDK.
5) Store all your images in the usual *assets/sorting* directory (all the files for all your sorting games should be in the same folder). This is also were the xml file should be stored. Make sure that the xml file for your new sorting game is called *sortingImages2.xml*.
6) You have now two sorting games! The first one saved as *sorting1.swf* (using *sortingImages*.xml) and the new one as *sorting.swf* (using *sortingImages2.xml*). Now you can easily add them both to the playlist or create a third minigame!

4. Possible Extensions

The new minigame media type is registered in XIMPEL ver. 1.7. As the new version 2.0, with youtube support, is launched, the possibility of adding minigames should be also included.

Minigames use a local connection to send a message to the main application with the score and the status of the game. If the user does not finish the game, they can still proceed to the next item in the playlist, just that the score will not be updated. It is worth considering such an extension, that in some cases users are not allowed to play the next media item before they have completed the minigame.