database for multimedia, macromedia applications
authoring systems

Robert KRÓL[*], Jan PIECHA[*,**]

THE EXAMPLE SHELL FOR MULTIMEDIA DATABASE MANAGEMENT

The multimedia applications are usually developed within various graphical interfaces. The application user expects to use rich formats for graphics, animation and various audio effects. Many efforts have already been done on this field. Anyhow new challenge one can observe that concerns the Internet applications [1], [2]. The paper shows the platform unifying the multimedia databases development, available on Wide Area Network servers.

1. INTRODUCTION

The user expectations of controlling the interaction within the database load presents many challenges to those who try to design these systems within educational settings. The fundamental data-unit called lesson consists of parallel layers of information that is selected by the user according to his private expectations of the data load. This way the presentation units are put into an individual structure of the application.

Let us consider the application composition consisting of several units (lessons) with several frames in each unit. In most cases they are available linearly - frame by frame, however many examples can be found where this linearity will not satisfy the user. The interaction drivers allow going through various units of information with several conditional solutions for additional examples, repetitions or interactive exploration of the application unit contents.

The conversation facilities of the multimedia database (courseware) interfaces are today main conditions that are expected by the application user. The application toolboxes for the presentation resources development are available as programming environment [2,3,4]. Many developers of this application kind prefer to use a general-purpose language to design an application.

Professional computer researchers provide the application software developers, with problem oriented design kits simplifying the application development processes.

The example proposal of the design platform environment has been presented in this paper. The application development shell is overworked in Macromedia Authoring package with several extensions to SQL.

[*] University of Silesia, Institute of Informatics, Dept. of Computer Systems, ul. Będzińska 39, Sosnowiec, Poland
[**] Technical Silesian University, Dept of Transport Informatics, ul. Krasińskiego 13, Katowice, Poland

The application designer uses several standards of:
- the application organisation,
- frames structure and interactions,
- the answer validation,
- the repetition procedures.

2. THE APPLICATION MODES

The whole application structure keeps a track of conversation entities going through the application units. The Multimedia Application Management Shell (MAMS) has been developed for unification of a presentation database. The third mode of the application structure has been presented in Fig.1.

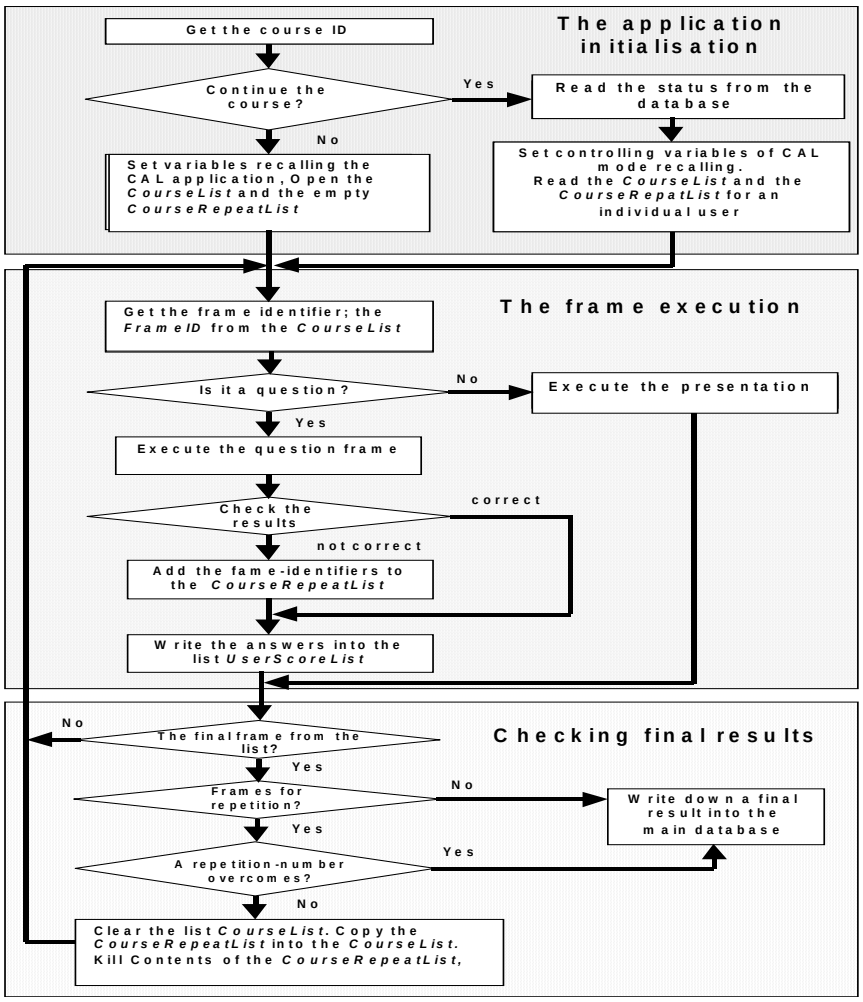

Fig.1. The application flowchart

The application user can select:

(a) a linear presentation mode, without any repetition,

(b) a full size repetition mode without any validation processes,

(c) a repetition mode of a selected part of the application, according to MAMS interactions (validation) results.

The repetition engine (for mode c) works as follows:

- a list *CourseList* contains identifiers of presentation and question frames,
- the *CourseRepeatList* contains frame identifier for repetition in a next presentation cycle, after the presentation is completed the current *CourseList* is cleared and the *CourseRepeatList* is moved into the *CourseList*.

The given answer is compared with a pattern in table *QuestionFrames*, then validated by several formulas. The table *ScoreLimit* keeps validation values thresholds.

3. THE PROGRAMMING SHELL STRUCTURE

The grouping icon *Course* (Fig.2) controls the unit set of the application.

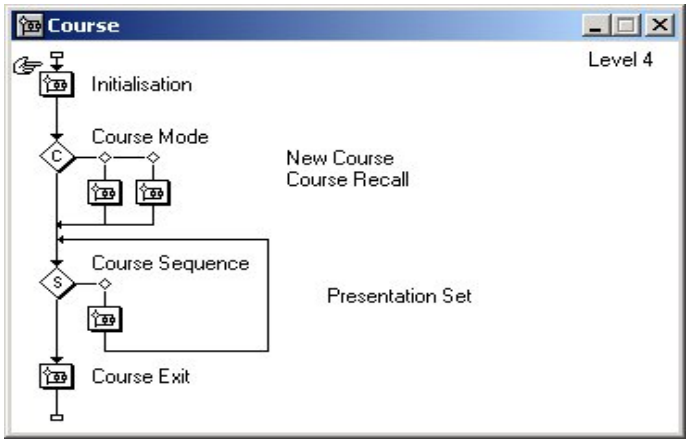

Fig.2. Icon Course

The *Initialisation* icon contents define:

- a starting point of the application, when it runs from the beginning,
- a restart conditions, when the application runs from a breakpoint.
  When variables: *ApplicationMode = 1*, the course starts from the beginning
  ApplicationMode = 2, the course runs from a breakpoint.

The application is running under variable values from unit *CourseMode* recorded in icons *NewCourse* and *CourseRecall*.

The *NewCourse* controls the courseware by initialisation variables:

- *Loop = 1*, for at least one full cycle of the course - to repetition number limit (defined in table *Courses*); the *Presentation Set* is executed until variable *Loop>=1,*
- *CourseLevel*, defines the current level of the course (in table *Courses*),
- *CourseCurrentRep*, indicates a current number of repetition,
- *RepetitionMode*, defines the repetition mode (read in table *Courses*),
- *Save* / *Break*, controls the application restart from the break point,
- *UserScoreList* contains variables list with results of the user interactions,

- *CourseRepeatList*, is keeping the list of frame IDs that are moved to the repetition list. The icon *NewCourse* contains the list *CourseList*, with two tables *PresentationFrames* and *QuestionFrames*, where identifiers of frames for current lesson have been given.
- *CourseIndex* – is a current frame's number of the lesson (in frames sequence),
- *CourseEndIndex* – the highest index of frame in the *CourseList*,
- *UserGoodAnswer*, *UserWrongAnswer* – variables that assign the answer result.

When the answer is positive the value of variable *UserGoodAnswer* is increased by *1*. When the answer is negative this variable is equal to *0*.

The variable *UserWrongAnswer* is increased by *1* in case the answer is wrong. For good answer this variable is equal to *0.* During the lesson execution both variables are compared with variable *CourseChangeLevel*. When the lesson runs in a multi-level mode the variable *CourseChangeLevel > 0*, then these vale is used for the lesson current level definition:

– for increasing the lesson level:

```
if (CourseChangeLevel>0) & (UserGoodAnswer=CourseChangeLevel) then
        UserGoodAnswer:=0
        if (CourseLevel<CourseMaxLevel) then
        CourseLevel:=CourseLevel+1
end if
```

– for decreasing the lesson level:

```
if CourseChangeLevel>0 & UserWrongAnswer=CourseChangeLevel then
        UserWrongAnswer:=0
        if CourseLevel>CourseMinLevel then
        CourseLevel:=CourseLevel-1
end if
```

- *QF_Frame* – contains an index of the lowest presentation frame number, that belongs to the current question (eg.: the frame Q1 is predicted by P1 and P2 then *OF_Frame=1* for P1).

The answer is compared with pattern available in table *Questions*. When the answer value is below the adequate value from the table *Questions* then the relevant frame identifier is put into the list *CourseRepeatList* and the *QF_Frame* value will be equal to value *CourseIndex*. (*QF_Frame=CourseIndex*).

For *CourseMode* = 2, the full cycle of repetition is established and icon *CourseRecall* is executed. The variables discussed above are set to values that are preserved in the course status (in the *UserStatus*).

The icon *Presentation Set* (Fig.3) collects all attributes of the presentation sequence, as:

- frames ID's selected for current application,
- calls, searching frames on server, when they are not found on workstation,
- the answers results for checking and copying frame indexes to *CourseRepeatList*,
- a present status of the course flow that is used for variable *Loop* modification, according to a selected repetition mode.

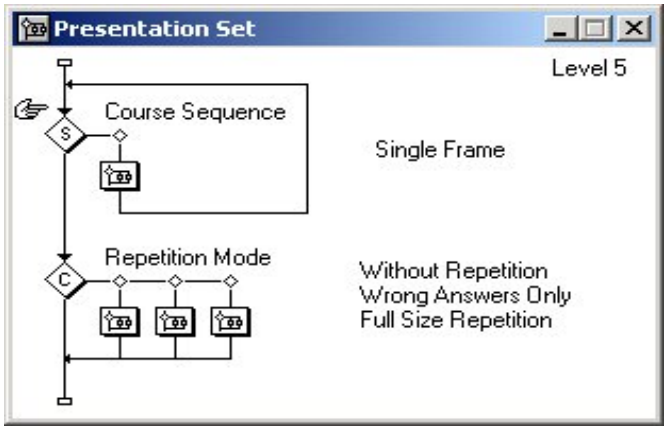

Fig.3. The icon Presentation Set

The icon *PresentationSet* works together with the icon *CourseSequence* (Fig.3) that controls the course execution, in accordance with the *Loop* variable value (for *Loop>=1*).

The icon *CourseExit* makes calculations of a final score within the interaction, displays the message of the result and sends the result into the database of the network (a total result and the answers score for every given question).

The icon *SingleFrame* (Fig. 4) performs a single application frame. The frame identifier is obtained from *CourseList* according to a current index – *CourseIndex*. After the frame is presented the *CourseIndex* is modified (increased or decreased by 1) to make the step forward or backward (in accordance with the answer result or with the navigation buttons).

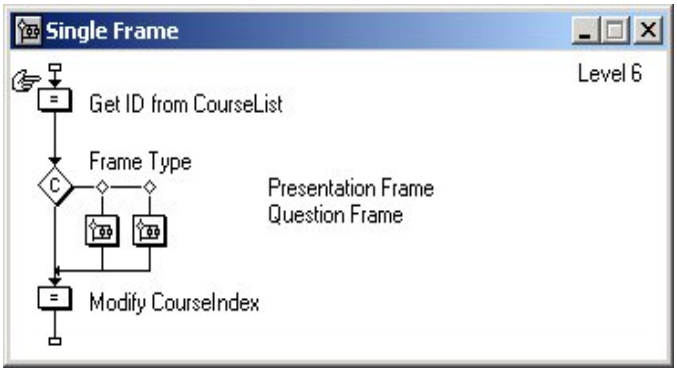

Fig.4. The single frame icon

The *SingleFrame* icon sequence goes ahead until the variable *CourseIndex* is not bigger than variable value *CourseEndIndex*. In the end of presentation the value of variable *RepetitionMode* in icon of the same name - *RepetitionMode* is checked. Then one of the application route is chosen: *Without Repetition*, *Wrong Answers Only*, *Full Size Repetition* – when it is 1,2 or 3 respectively.

In the icon *Get ID from CourseList* the frame identifier of successive frame is read and the frame mode is checked (presentation or question frame). The variable *Branch* is equal to *1* or *2* for presentation or for question frame, respectively:

```
ss:=GetWord(1;CourseList[CourseIndex])
if ss="p" | "P" then
      FrameID:=GetNumber(1;GetWord(2;CourseList[CourseIndex]))
      Branch:=1
end if
if ss="q" | "Q" then
      QuestionID:=GetNumber(1;GetWord(2;CourseList[CourseIndex]))
      Branch:=2
end if
```

The data found in tables *Presentation Frames*, *Question Frames* and *Supplementary Library Files* modify the list of the application files. Next the icon *Jump*, from the above icons, is executed (fig. 5).

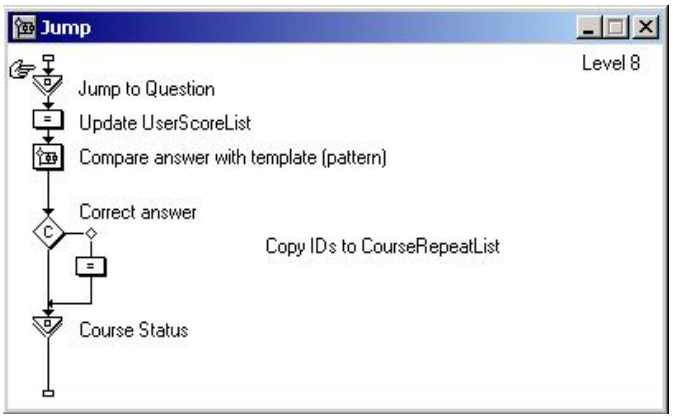

Fig.5. The Jump Icon.

After the answer result is checked the program returns into the icon *Jump*, where calculation in *Update* the *UserScoreList* is performed. The *UserScoreList* is modified by components that assign the given question, by variables: *QuestionID*, *UserId*, *UserScore* and *CourseLevel*.

The remaining information concerns:

- a current repetition-number of question, indicated by variable *CurrentRepNo*,
- a date and an hour of the application start-up,
- a date and an hour of the answers noticed in the application status (the answers duration).

– The variable *StatusString* consists of:

StatusString:=”Question ID:"^QuestionID^"\t"^UserScore^" Level: "^CourseLevel^" CurrentRepNo :"^CurrentRepNo^" DateStart: "^QDateStart^" TimeStart:"^QTimeStart^" DateStop: "^QDateStop^" TimeStop:"^QtimeStop

– The ***UserScoreList*** modifies: AddLinear(UserScoreList;StatusString)
– The ***UserStatus*** is updated using the SQL commands:

UPDATE UserStatus SET currentrep=CourseRepNo, score=UserScore, userscorelist=StatusString, usergoodanswer=UserGoodAnswer, userwronganswer=UserWrongAnswer WHERE courseid=CourseID AND userid=UserName

The variable StatusFile define operations:

StatusFile:=FullDate^"\t\t"^QTimeStart^"\t\t\tJump To Question Name: "^QuestionTitle^" Question File Name: "^QuestionFName^"\r"

– The variable value *StatusFile* is written into the file by:

AppendExtFile(getOSDirectory()^"\\Temp\\"^UserLogFName;StatusFile)
StatusFile:=FullDate^"\t\t"^FullTime^"\t\t\tReturn From Question Name: "^QuestionTitle^" QuestionFName: "^QuestionFName^"\r"
AppendExtFile(getOSDirectory()^"\\Temp\\"^UserLogFName;StatusFile)

– The icon *Compare answer with template pattern* analyses several results of the answer score:

```
if UserScore>=ScoreLimit then    {for proper answer}
  UserGoodAnswer:=UserGoodAnswer+1
  UserWrongAnswer:=0
  QF_Frame:=1
  Branch:=0
else                             {for not satisfying answer}
  UserWrongAnswer:=UserWrongAnswer+1
  UserGoodAnswer:=0
  Branch:=1
end if
```

– The presentation level is defined:

```
{for higher level}
if (CourseChangeLevel>0) & (UserGoodAnswer=CourseChangeLevel) then
  UserGoodAnswer:=0
  if (CourseLevel<CourseMaxLevel) then
CourseLevel:=CourseLevel+1  {the presentation level grows}
end if
{for lower level}
if CourseChangeLevel>0 & UserWrongAnswer=CourseChangeLevel then
  UserWrongAnswer:=0
  if CourseLevel>CourseMinLevel then
CourseLevel:=CourseLevel-1 {the presentation level drops down}
end if
```

– The *UserScore* is compared with the value of *ScoreLimit*.
Every question describes its unique number in *Question Frames* table. For example: question with identifier *1* in its field of *ScoreLimit* has value 50. Let the answer (*UserScore*) is equal to 60 then comparison result will satisfy the validation threshold.

The variable value *UserGoodAnswer* is equal to 1 for the answer result above the assumed level. It is 0 if the comparison is not satisfying the assumed criterion.

For variable *CourseChangeLevel* = 1 the course level goes higher, the variable *CourseLevel* is modified respectively to the given limit. Similarly, for not satisfying answers the application level drops down, and more frames are presented. For one level application, the variable *CourseLevel* = 1; stays the same.

For not satisfying answers the variable *Branch* = 1. Then the operation *Copy Ids to CourseRepeatList* is performed, where the question-frame identifier and its predicting frames identifiers (down to the lower question-frame) are included into the *CourseRepeatList*.

Example:

Let the *CourseList[„P1”,”P2”, „Q1”, „P10”,”P20”, „Q 10”]* the user did a not satisfying answer in frame „Q10”. Then identifiers of frames „P10”, „P20”, „Q10” are included into the *CourseRepeatList*.

Finally in the icon *CourseStatus* the course results are noticed. The SQL command enters the course results into the record of *UserStatus*,

with values of variables:

CourseIndex, *CurrentRepNo*, *CourseLevel*, *UserGoodAnswer*, *UserWrongAnswer*,

and lists contents:

CourseList, *CourseRepeatList*, *UserScoreList*.

After a whole presentation cycle the *RepetitionMode* is read. *0* indicates none repetition mode (Fig.6). The *Dec Loop* controls a variable *Loop*, that means the course presentation goes for the last time. The decision icon *Course Sequence* drives the application to icon *Course Exit*.

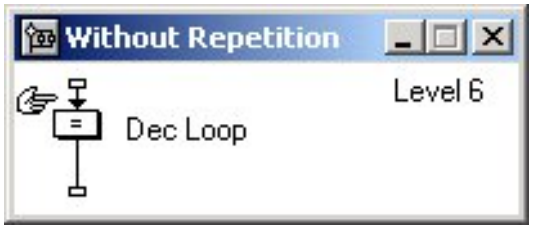

Fig.6. None repetition path

For *RepetitionMode = 1* the *Wrong Answers Only* icon is executed (fig. 7). It means that only part of the application is repeated.

Fig.7. The repetition path for Wrong Answer Only

For *Wrong Answer Only* the variable value of *CurrentRepNo* increases by 1 then it is compared with the variable value of *MaxRepNo* (repetitions limit). When this value is smaller then or equals the assumed limit the repetition sequence runs once more.

The variable value *Loop* decreases and the *CurrentRepNo* increases, than all variables of *CourseList* are cleared. After copying the *CourseRepeatList* to current *CourseList* the repetition runs from the beginning.

When the *CourseRepeatList* is empty means that none of frames will be repeated; all answers were given properly. The variable *Loop* is set to *0*, the course goes to the exit.

For *RepetitionMode = 2* means that the application goes through the *Full Size Repetition* icon (fig.8).

Fig.8. The repetition path for full size of the repetition.

The *Full Size Repetition* icon makes a comparison of variable values *CurrentRepNo* and *MaxRepNo* and a whole application is repeated (*Copy CourseRepeatList To CourseList* means rewriting the *CourseRepeatList* to *CourseList* – previously cleared). The operations *IncCurrentRepNo & Dec Loop* are performed.

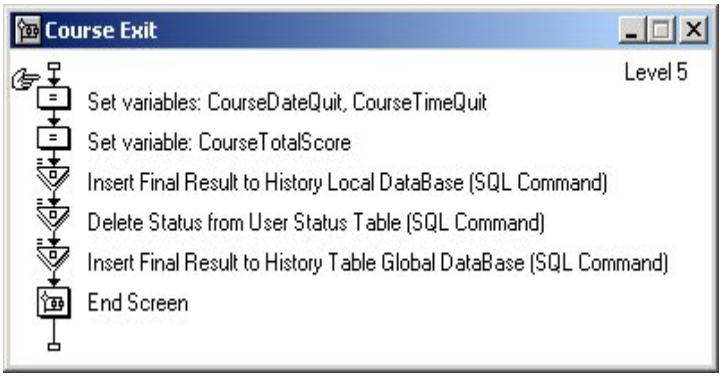

Fig.9. The course exit

In *Course Exit* (*Fig.9*) several closing procedures are executed, modifying a local and global database of the application.

4. CONCLUSIONS

The above discussion shows the platform that simplifies the macromedia applications development. The control mechanisms for management of the user interactions within the database

load (generally dedicated for learning processes) can be modify into one of three modes (*a, b or c*). It can also be used for general-purpose databases exploration, especially round multimedia files working in mode *a* or *b*.

BIBLIOGRAPHY

[1] KRÓL R. The access method to internet databases developed within the macromedia environment. Journal of Medical Informatics and Technologies. Vol.5, Nov. 2000, ISBN 83-909518-2-7, pp. IT 91-98.
[2] PIECHA J.,KRÓL R., PAWEŁCZYK P. A network node management shell for macromedia applications. Proc. of Conf. KOSYR 2001, ISBN 83-911675-2-6, pp. 493-499.
[3] PIECHA J. The programmable shell for multimedia applications development. *Journal of Applied Computer Science*, Vol.7, No 2, pp.31-43, ISSN 1507-0360, Łódź 1999.
[4] PIECHA J. The Intranet Databases and some Approach Troubles into Multimedia Files. Proc. Int. Conf. *Computer Based Learning In Science - CBLIS'99*, Enschede, the Netherlands, 1999, G7.