Same Same But Different – Comparing Rendering Environments for Interactive Digital Objects

M. Guttenbrunner[1,2], J. Wieners[3], A. Rauber[1], and M. Thaller[3]

[1] Vienna University of Technology, 1040 Vienna, Austria
{guttenbrunner,rauber}@ifs.tuwien.ac.at
[2] Secure Business Austria, 1040 Vienna, Austria
mguttenbrunner@sba-research.org
[3] University of Cologne, Cologne, Germany
{jan.wieners,manfred.thaller}@uni-koeln.de

Abstract. Digital cultural heritage in interactive form can take different shapes. It can be either in the form of interactive virtual representations of non-digital objects like buildings or nature, but also as born digital materials like interactive art and video games. To preserve these materials for a long term, we need to perform preservation actions on them. To check the validity of these actions, the original and the preserved form have to be compared. While static information like images or text documents can be migrated to new formats, especially digital objects which are interactive have to be preserved using new rendering environments.

In this paper we show how the results of rendering an object in different environments can be compared. We present a workflow with three stages that supports the execution of digital objects in a rendering environment, the application of interactive actions in a standardized way to ensure no deviations due to different interactions, and the XCL Layout processor application that extends the characterized screenshots of the rendering results by adding information about significant areas in the screenshot allowing us to compare the rendering results. We present case studies on interactive fiction and a chess program that show that the approach is valid and that the rendering results can be successfully compared.

Keywords: Digital Preservation, Rendering, Preservation Planning, Characterization, Emulation, Image Segmentation.

1 Introduction

As more and more representations of our cultural heritage are recreated in digital and interactive form (e.g. interactive models of buildings [1], nature areas [2] or ancient cities [3], we have to ensure, that these representations can be accessed in future rendering environments as well. Also born digital materials like interactive art and video games have to be preserved as part of our digital cultural heritage.

Most recent digital preservation projects concentrated on migration of documents as a main strategy for preserving images and documents. Automatic evaluation of the

M. Ioannides (Ed.): EuroMed 2010, LNCS 6436, pp. 140–152, 2010.

results of migration processes, e.g. for evaluating the validity of tools, compares characteristics of the original and the migrated files. Previous work on characterization of digital objects made clear, that the separation between the properties of digital objects which reside in their persistently stored form and the properties which are inherent in the rendering software is not always sufficiently clear. While for some types of objects most of the information about the rendering can be found stored in the object (e.g. text files, image files) and the rendering software just interprets these properties. On the other hand e.g. 3D objects stored in the resource files of a computer game describe the object but are put into context with the game environment, the player actions, the view-point or lightning only due to the game logic (=the rendering software). In this case it is not possible to deduce the rendering properties from a stored version of the object. Having the object rendered and comparing the outcome of rendering processes from different environments (e.g. the original and an emulated environment) makes it possible to evaluate, if the interaction properties, i.e. the reaction of a digital object to interaction, stay intact. Similar principles apply also to all static objects, as preservation always needs to focus on the intellectual object (the object as conceived by the observer), rather than the stored representation, which always has to include the view path, requiring us to compare rendered forms.

In this paper we describe how the XCL Layout Processor [4] as an extension to the Extensible Characterization Language (XCL) [5] is used to compare different renderings of a digital object. We present a workflow for applying interaction to the same digital object in different rendering environments and measuring the results of comparing significant states of interactive objects. We describe how interaction can be applied to digital objects by using a tool that captures and replays input to the different environments and how the outcome can be compared by taking a screenshot at a certain point in execution time. Using this approach we show how the effects of interaction on a digital object can be measured comparing screenshots of the environment taken from a target state or different intermediary states of the object. This allows for automatic comparison of rendering results and thus supports the evaluation of rendering environments.

The paper is structured as follows. First we give an overview on related work on the subject. In Section 3 we explain how we apply interaction and when we measure the results. Then Section 4 shows how the screenshots we take as a result of the rendering process are analyzed and mapped in the characterization language. In Section 5 the workflow for comparing different rendering environments is shown and in Section 6 we present a case study on interactive fiction and a chess program. Finally the conclusions are presented and an outlook to future work is given.

2 Related Work

The UNESCO guidelines for the preservation of digital heritage [6] list migration [7] and emulation [8] [9] as the main strategies for digital preservation. Migration is the strategy to keep a digital object accessible by converting it to a non-obsolete format, while emulation refers to the capability of a device or software to replicate the behaviour of a different device or software. With emulation the object is kept accessible in its original environment. Previous research has concentrated on methods for evaluating the

effects of migration on documents. A preservation planning workflow is described in [10] and allows repeatable evaluation of preservation alternatives. An implementation of this workflow has been done in the preservation planning tool Plato [11], utilizing automatic characterization of migrated objects with tools like Droid [12] to identify files. The significant properties of migrated objects can be compared automatically using the eXtensible Characterisation Language (XCL) [5] to measure the effects of migration on the object.

While the preservation planning tool can be used to compare rendering environments as shown in a case study in [13], the comparison has to be done manually. In [14] the information contained within a file is distinguished from the rendering of this information. Thaller shows that the rendering process can produce different outcomes for the same file if a different view-path for rendering a digital object is used. This makes it necessary to not only compare properties of the file but also properties of the outcome after a rendering process (e.g. output on a screen).

In [15] van Diessen describes the view-path as "a full set of functionality for rendering the information contained in a digital object". The view-path contains the hardware and all the secondary digital objects needed to render an object along with their configuration. It is possible to use different view-paths to display the same object. On various hardware-configurations different operating systems (e.g. WinXP, Linux) can be used to run different applications (e.g. Word, Open Office) to render the same object (e.g. Word-97 Document).

In [16] the levels on which significant properties can be extracted are described and it is shown, what continuity has to be considered when extracting properties with regards to a defined view-path and standardized interaction to expect the same results. In [4] Wieners describes how the XCL Layout Processor was developed as an extension to the XCL tools to extract significant coordinates of areas in an image such as a screenshot of a rendered digital object. For the image segmentation the screenshots of the rendering outcomes are first reduced to monochrome images to facilitate the edge detection. Otsu's global thresholding method as described in [17] is used for this. Then these monochrome images are segmented using an image segmentation algorithm described in [18].

3 Identifying Interaction Properties

Previous tests with different interactive digital objects presented in [13] and [16] made it clear, that some significant properties concerning interaction have to be defined for every interactive digital object. It is necessary to research to what types of input (e.g. certain keys on a keyboard, mouse pointer movement or mouse clicks, joystick input) a digital object responds, but also to the timing in which this input has to occur. While the same interactive response of an action game relies very strong on the exact same timing of the input, in interactive fiction like text adventures or point-and-click adventures the timing is usually not as critical.

By having a user manually interact with the digital object and recording the user input to the digital object as well as the times relative to the object execution start we are able to replay the same input automatically. Using a separate tool this can be done independent of the object execution environment (e.g. a video game rendered in different

emulation environments). This approach allows us the use not only on emulation environments but also on other execution environments (e.g. source ports of a software or using game engine interpreters like ScummVM[1] or Frotz[2] for video games). By defining an "end point" where the user finishes the input actions and the interactive object finishes responding to these actions we can take a screenshot of the rendering at a certain point of execution both when recording the user action but also after replaying the user action to a different (or even the same) rendering environment. In the next section we describe the process of analysing the image and placing the significant properties of identified segments in the image in the corresponding XCDL of the screenshot.

4 Image Segmentation and Mapping in XCL

The central analysis task of the method described in this chapter is to identify specific regions in the rendered digital object, which can – in a final step – be compared with the rendering of the same object, using another rendering environment. Those specific regions reflect characteristic layout properties: regions with a high frequency of pixels that could refer to a significant area. To identify and isolate such regions of interest in the prepared, cut to size and binarized image, the image is segmented by the Efficient Graph-Based Image Segmentation Algorithm as presented in (Felzenszwalb, 2004).

For each black pixel in the binarized image, the algorithm determines the affiliation of the processed pixel to a specific region by using three different parameters that influence the operation-mode of the segmentation algorithm:

- σ (Sigma) indicates how strongly the image is smoothed by the Gaussian filter. The higher σ, the more the image gets smoothed.
- k influences the "scale of observation" : the larger k, the larger the components identified by the segmentation algorithm are.
- min determines the minimum component size: the smaller the size of min, the more objects will be identified by the segmentation algorithm.

To facilitate comparison between two screenshots, the proposed solution sets the upper leftmost pixel and the bottom rightmost pixel of a layout-screenshot in relation to the pixel dimensions of the image by dividing both pixel values by the width, respectively the height, of the processed image. Finally, these relative values are embedded into XCDL files, which are connected to the screenshots, to enable a comparison of the objects through the XCL Comparator (Becker et.al., 2008b).

The structure of the input XCDL is supplemented by a new property with a unique identifier: A new property with the id "p15381" (fixed value) is inserted into the XCDL and represents the significant points of the isolated objects in the screenshot file through different valueSet Tags. It is inserted after the XML header, the XCDL Schema and the normData part of the XCDL file, and is visualized in the code snippet shown in Figure 1.

---

[1] ScummVM - Graphical point-and-click adventure game engine interpreter: http://www.scummvm.org/

[2] Frotz – Game Engine Interpreter for Infocom and other Z-machine games: http://frotz.sourceforge.net/

```
<property id="p15381" source="raw" cat="descr">
   <name id="id9998">significantCoordinates</name>
   <valueSet id="i_i1_i2xx_s1_1">
      <labValue>
         <val>0.118727 0.113586 0.232558 0.335189</val>
         <type>rational</type>
      </labValue>
   </valueSet>
</property>
```

Fig. 1. Code snippet of XCDL enhancement for significant coordinates of identified areas

The four floating point numbers in the first valueSet of the generated XCDL (<val>*leftupper_X leftupper_Y rightlower_X rightlower_Y*</val>) represent the relative pixel position of the significant points, identified through the segmentation process. Therefore, the value *leftupper_X* (0.118727 in the example in Figure 1) indicates the position of the relative x coordinate of the topmost left pixel of the identified object; the second value, *leftupper_Y*, (0.113586 in the example in Figure 1) indicates the relative y coordinate of the topmost left pixel of the identified object. The next two values refer to the relative x and y coordinates of the bottommost right pixel values of the identified object.

An application that accomplishes the described tasks was created as the XCL Layout Processor. A screenshot of the application in use for comparing screenshots of two renderings of a text document (on the one side rendered with Microsoft Word 2007, on the other side rendered with Adobe Reader) can be seen in Figure 2. The screenshot as well as the corresponding XCDL characterization file are loaded into one region of the application of the application's GUI. The XCDL is then enhanced as described above with the significant coordinates of the recognized areas. For a visual comparison a second screenshot and XCDL file can be loaded into the second region and processed likewise.

5 Applying Interaction to Digital Objects and Comparing Rendering Results

To evaluate if the interactive properties of a digital object are preserved properly in a different rendering environment than the one originally intended for the object it is necessary to ensure that the same kind of input is applied to the object at the same point in time. By keeping the secondary digital objects in the view-path for the object as well as external influences like user input unchanged, differences in the rendering are caused by a change in the rendering environment.

A conceptual workflow for comparing the interactive properties of a digital object in different rendering environments is drafted below. It consists of 3 stages with different steps in the stages as shown in Figure 3.

Stage 1: Recording the Original Environment. In this stage the user actions are recorded in the original environment and screenshots of the original rendering process are taken as "ground truth" against which other environments are evaluated. The following steps are followed:

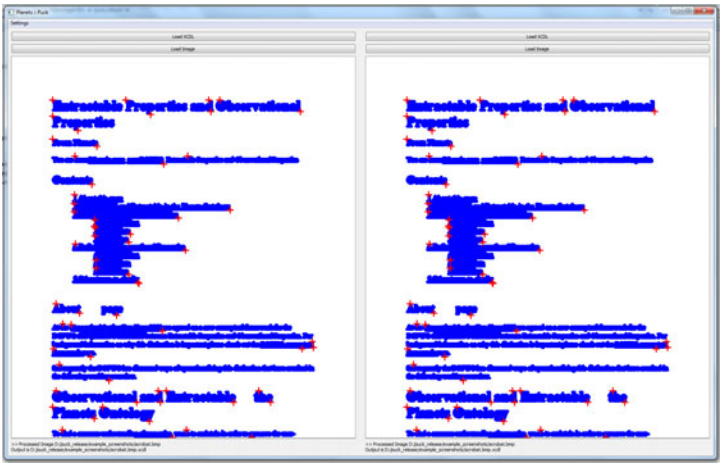

Fig. 2. Comparison of two renderings of a text document in the XCL Layout Processor. Identified areas are shown in blue, identified coordinates as red crosses.

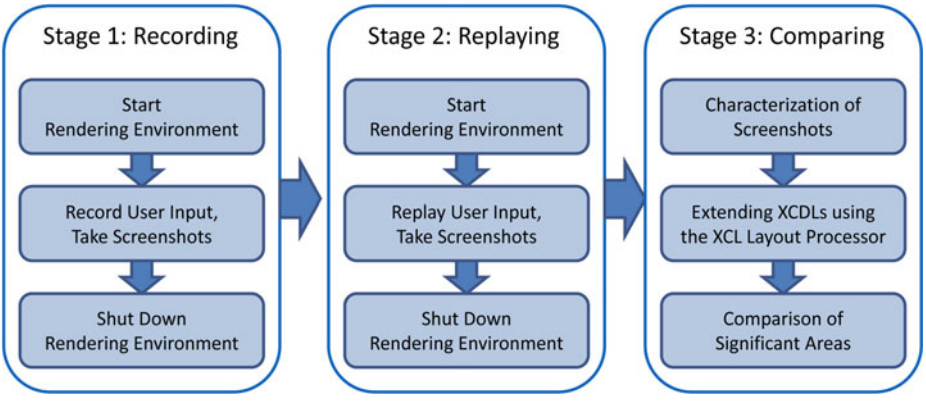

Fig. 3. Conceptual workflow for comparing rendering results of interactive digital objects

1. start the rendering environment with the digital object
2. record the user actions (e.g. in a standardized XML-Format) and take screenshots at predefined intervals or one screenshot after a certain amount of time
3. shut down the rendering environment

Stage 2: Replaying. In this stage the recorded user actions are applied to the alternative rendering environment. The same settings for screenshot interval etc. are used as when recording in the original environment. These steps in this stage are carried out for every alternative rendering environment that is evaluated:

1. start the rendering environment with the digital object (e.g. different emulation environment)

2. replay the user actions from a recorded session and take screenshots at the same predefined intervals or one screenshot after a certain amount of time as in the original recording session
3. shut down the rendering environment

Stage 3: Comparing. Finally in this step the rendering process are compared. Therefore the screenshots need to be characterized and the following steps to be taken to compare the screenshots taken during the rendering processes. The steps in this stage have to be repeated for every alternative rendering environment that is evaluated.

1. Characterization of the screenshot images
2. Extending the XCDL for the screenshots with the coordinates of the identified significant areas using the XCL Layout Processor
3. Pair wise comparison of the screenshots taken at the same object execution time using the XCL comparator to identify differences in the rendering

Using the workflow drafted above, we have established a formalized way to compare the rendering results for the same digital object in different rendering environments. The workflow can be implemented in a tool to support the automatic comparison of rendering of interactive digital objects.

6 Case Study on Interactive Objects

We evaluated the proposed workflow for testing interactive properties of digital objects by comparing the rendered outputs on two case studies. We used the XCL comparator to compare XCDL files of the rendering outcome screenshots. These XCDL files were extended by coordinates of significant areas using the XCL Layout Processor introduced in Section 4.

For the first case study we used the video game "The Secret of Monkey Island". This game was developed by Lucasfilm Games and released on various platforms from October 1990. The game is a typical point and click adventure game, thus not requiring interaction timing that is exact down to millisecond level. It was also chosen because conceptually different rendering engines exist, that are usable for digital preservation purposes:

- ScummVM: a game engine interpreter available for various non-obsolete platforms
- Emulation/Virtualization of the original hardware (e.g. DOSBox[3])

Various tools for recording interaction and replaying it in the form of a macro were evaluated but failed especially in recording/applying interaction in the game running in the original environment (as a Full-Screen DOS-application). As no tool for automating the process described in the previous chapter exists yet, the steps were performed manually. First the game was executed on a workstation running Windows XP. The movements of the mouse pointer as well as the keystrokes and times of these events were manually noted and the game was played up to a point where the character in the game that is controlled by the player enters the SCUMM Bar and talks to

[3] DOSBox: http://www.dosbox.com/

pirates sitting on a table telling "I want to be a pirate." At that point in the game a screenshot was taken.

Subsequently the same actions were then performed by trying to replicate the same timing by running the game under a virtualized environment using DOSBox 0.72 and using the ScummVM 1.1.148874 Engine (using also the game data files from the DOS Version of the game). For ScummVM an unscaled rendering replicating the "EGA" settings that were similar to the options of the real DOS-Version of the game were used.

The screenshots taken in the three different rendering environments were then characterized using the XCL tools. Then the XCL Layout Processor was used to binarize and segment the screenshots and extend the XCDLs of the images. Figure 4 shows a screenshot from the original DOS-version of the scene defined as "endpoint" for this scenario on the left. On the right the same screenshot as segmented by the XCL Layout Processor is shown. The image is binarized to black/white to identify areas. Different greyscales present in the segmented image are just a visualization of the different segments in the picture. The following values were used for the segmentation algorithm: σ=0.8, k=1000 and min=100. Figure 5 and Figure 6 show the segmentation for the screenshots of the other two rendering environments of the game. A visual comparison of the segmentations shows, that the game running in the DOSBox environment is segmented to very similar areas as in the original version, whereas in the ScummVM version a lot more differences can be found.

The XCL Layout Processor enhances the original XCDL which was created by using the XCL extractor on the screenshots taken in the different environments. Table 1 shows the number of significant areas identified per screenshot.

Table 1. Significant areas recognized in different renderings of the same scene in "The Secret of Monkey Island"

Rendering Environment	Original	ScummVM	DOSBox
Significant Areas in XCDL	**62**	**66**	**62**

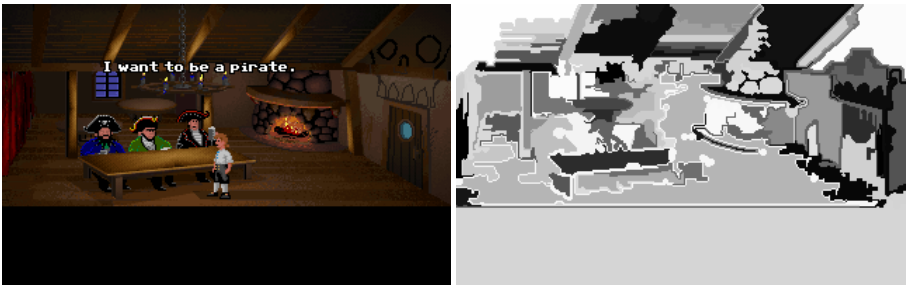

Fig. 4. Screenshot of original DOS-Version of "The Secret of Monkey Island" (left). Significant areas in the same screenshot as a result of binarization and segmentation are shown on the right.

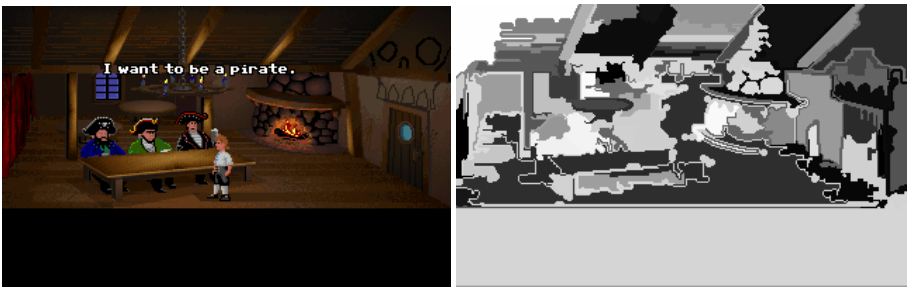

Fig. 5. Screenshot of "The Secret of Monkey Island" running in the DOSBox Environment (left). Significant areas in the same screenshot as a result of binarization and segmentation are shown on the right.

Fig. 6. Screenshot of "The Secret of Monkey Island" using ScummVM as a rendering engine (left). Significant areas in the same screenshot as a result of binarization and segmentation are shown on the right.

Using the XCL comparator we then compared the XCDLs of the different screenshots. The comparator reported failure for both comparisons. On closer inspection of the XCDLs of the screenshots the following facts were observed:

- The original and the DOSBox version differed in significant areas that were recognized (e.g. a fire burning in the chimney, pirate's heads moving). The reason are animations in the picture which lead to slightly different images and thus to different areas that are recognized by the segmentation algorithm. Blocks not animated and without other animated blocks overlapping do have the same coordinates.
- The original and the ScummVM version differed in that the colour spaces of the screenshots were different. While the original version was rendered in an 8bit colour space (palette mode), ScummVM rendered the image in a 24bit colour space (true-colour mode). Even though the number of significant areas was coincidental equal, the coordinates differed throughout all areas.

Based on these differences we can draw the following conclusions:

- Timing of screenshots together with the input is important, as animations in interactive dynamic digital objects that occur continuously (e.g. changes in the environment or in small movements of characters) changes the screenshot and thus

leads to different coordinates of significant areas and also to different areas that might be recognized as significant.

- This in turn leads to the fact that the values for the segmentation algorithm have to be balanced accordingly to detect the same significant areas even when slight changes in objects occur. The algorithm has to be configured sensitive enough to recognize enough areas to compare two screenshots and detect differences, but insensitive to minor differences in an image that lead to changes in recognizing a significant area as being exactly that.

To validate the outcome of the XCL Layout Processor on a digital object which would not pose the problems of animations and need an exact timing of key presses and screenshots taken, we made a second case study on the game "Chessmaster 2100" published for the DOS platform in 1988. Again the original software running in DOS was compared to the same program running in DOSBox in Windows XP. A few beginning moves in a game of chess were played with trying to keep the timing of the moves intact manually. The screenshots taken as well as the segmentations of the screenshots can be seen in Figures 7 and 8 respectively. For all the elements on screen to be correctly visible on the segmented image (e.g. the numbers left and below the board, all the figures in the squares) the following values were used for the segmentation algorithm: σ=1.0, k=1000 and min=100.

A first inspection of the images shows that the colour depth in DOSBox was increased to 8bit compared to 4bit in the original image. This is also reflected in the extracted XCDL of the images. Visually this also results in slightly different colour shades in the extracted images and is also reported when comparing the images using the XCL comparator as difference in the colour palette. Comparing the XCDL files enhanced with coordinates for significant areas by the XCL Layout Processor, we can see that the identified areas in the images are exactly the same in number (153 recognized areas) and coordinates.

If we compare the results to the case study on "The Secret of Money Island" we can see that depending on the digital object and on the fact that no animations change the image in the evaluated chess program, the timing of screenshots and interaction is less crucial and allows us to manually evaluate the rendering results for certain interactive digital objects like "Chessmaster 2100", thus confirming the validity of the approach of using the XCL Layout processor for comparing rendering outcomes after applying interaction to the object.

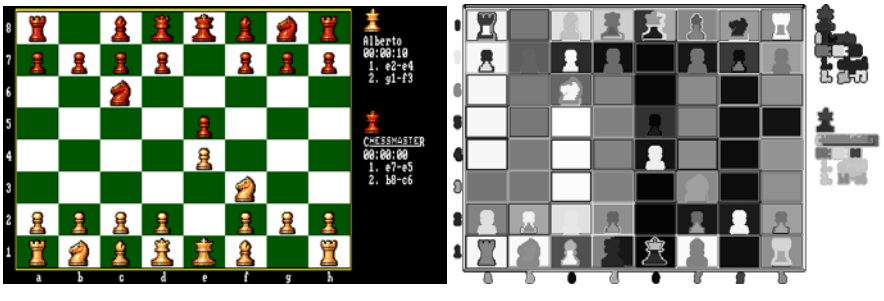

Fig. 7. Screenshot of "Chessmaster 2100" running under DOS on the left and the segmented screenshot showing significant areas on the right

Fig. 8. Screenshot of "Chessmaster 2100" running under DOSBox in Windows XP on the left and the segmented screenshot showing significant areas on the right

7 Conclusions and Future Work

The work presented in this paper showed an approach to identify characteristic objects of rendered digital objects based on screenshots in certain stages during the lifetime of the object. Identification of significant areas in the screenshot is done using pre-processing methods like cutting and reducing the image information through binarization and, finally, the Graph-Based Image Segmentation Algorithm. By comparing the rendering results of one or more pre-determined states during the runtime of a digital object that responds to user input it is possible to evaluate, if a digital object reacts to interactive actions as expected and in the same way over various different rendering environments. The conditions for the rendering must not change over different environments. Different lighting conditions or a different view-point in the new rendering environment will result in a change in rendering, thus producing a different image even though an object behaves similar in two different rendering environments.

We introduced a conceptual workflow for recording user interaction in an original environment along with screenshots along the path, with applying the same interaction and taking screenshots in the same points in execution in other rendering environments. Using the extensible Characterization Language (XCL) properties of the screenshots along with the identified significant areas in the images are compared to evaluate, if a rendering environment is creating the same result as the original rendering environment.

We carried out case studies on interactive fiction using the game "The Secret of Monkey Island" and one on the chess playing program "Chessmaster 2100". The result of the case studies showed:

- It is important to consider exact timing both of interaction but also of the time when the screenshot is taken, to compare the same rendering results, as changes in the image that do not occur due to interaction (e.g. animations of the game environment or characters) influence the result of the segmentation algorithm. If the resulting image is constant exact timing is less crucial.
- Identifying the accurate segmentation parameters (σ, k, min) for a certain digital object is crucial for correctly recognizing the significant areas over different

screenshots, especially if the rendering environments use different colour depth or image resolution for rendering the digital object.

The case study on "Chessmaster 2100" also confirmed that comparison of different rendering environments using the XCL Layout Processor and the XCL Tools can actually be used to evaluate if interactive properties of digital objects are preserved in different rendering environments by comparing rendering outcomes after applying interaction to the digital objects.

For future work it is necessary to implement the proposed workflow in a tool, as exact timing is not possible with manual interactions. Preliminary tests with various keyboard recording and screenshot tools also showed that depending on the environment it is not always possible to record/interact from outside the rendering environment, making it necessary to support the process inside the rendering environment (e.g. an emulator). Additional work has to be done on the segmentation process and the determination of the segmentation parameters for different types of digital objects.

Acknoledgements

Part of this work was supported by the European Union in the 6th Framework Program, IST, through the PLANETS project, contract 033789.

References

1. Foni, A., Papagiannakis, G., Magnenat-Thalmann, N.: Virtual Hagia Sophia: Restitution, Visualization and Virtual Life Simulation. Presented at the UNESCO World Heritage Congress (2002)
2. DeLeon, V., Berry, R.: Bringing VR to the Desktop: Are You Game? IEEE MultiMedia 2000, 68–72 (2008)
3. Maïm, J., Haegler, S., Yersin, B., Mueller, P., Thalmann, D., Vangool, L.: Populating ancient pompeii with crowds of virtual romans. In: Proceedings of the 8th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST 2007), pp. 26–30 (2007)
4. Wieners, J.: Extend the capabilities of the extraction tools to extract layout characteristics. PC/4 - D13. Internal Deliverable, EU Project Planets (2010)
5. Becker, C., Rauber, A., Heydegger, V., Schnasse, J., Thaller, M.: Systematic characterisation of objects in digital preservation: The extensible characterisation languages. Journal of Universal Computer Science 14(18), 2936–2952 (2008)
6. Webb, C.: Guidelines for the Preservation of the Digital Heritage. In: Information Society Division United Nations Educational, Scientific and Cultural Organization (UNESCO) – National Library of Australia, http://unesdoc.unesco.org/images/0013/001300/130071e.pdf
7. Slats, J.: The Digital Preservation Testbed - Migration: Context and current status. Whitepaper, National Archives and Ministry of the Interior and Kingdom Relations (2001)
8. Rothenberg, J.: Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation. In: Council on Library and Information Resources (1999), http://www.clir.org/pubs/reports/rothenberg/contents.html

9. Van der Hoeven, J., Lohman, B., Verdegem, R.: Emulation for digital preservation in practice: The results. International Journal of Digital Curation 2(2), 123–132 (2008)
10. Becker, C., Kulovits, H., Guttenbrunner, M., Strodl, S., Rauber, A., Hofman, H.: Systematic planning for digital preservation: Evaluating potential strategies and building preservation plans. International Journal on Digital Libraries (IJDL) (December 2009)
11. Becker, C., Kulovits, H., Rauber, A., Hofman, H.: Plato: a service-oriented decision support system for preservation planning. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL 2008), Pittsburgh, Pennsylvania, June 16-20 (2008)
12. Brown, A.: Automatic format identification using pronom and droid. Digital Preservation Technical Paper 1 (2008), http://www.nationalarchives.gov.uk/aboutapps/fileformat/pdf/automatic_format_identification.pdf
13. Guttenbrunner, M., Becker, C., Rauber, A.: Keeping the game alive: Evaluating strategies for the preservation of console video games. International Journal of Digital Curation (IJDC) 5(1), 64–90 (2010)
14. Thaller, M.: Interaction testing benchmark deliverable PC/2 - D6. Internal Deliverable, EU Project Planets (2008)
15. Van Diessen, R.J.: Preservation requirements in a deposit system. IBM/KB Long-Term Preservation Study Report Series, vol. 3, ch. 3 (2002), http://www-05.ibm.com/nl/dias/resource/preservation.pdf
16. Guttenbrunner, M.: Evaluating the effects of emulation environments on rendering digital objects. Internal Deliverable PP5/D2, EU Project Planets (2009)
17. Otsu, N.: A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics 9(1), 62–66 (1979)
18. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient Graph-Based Image Segmentation. International Journal of Computer Vision 59(2), 167–181 (2004)