Abstract

The goal of the research is creation of an adaptation mechanism for the delivery of three-dimensional content. The adaptation of content, for various network and terminal capabilities - as well as for different user preferences, is a key feature that needs to be investigated. Current state-of-the art research of the adaptation shows promising results for specific purpose and limited types of content but still it is not well adoptable for the massive heterogeneous environments. In this research, we present a method for transmitting adapted thee-dimensional contents to multiple target devices. This paper presents some theoretical and practical methods for adapting three-dimensional contents, which includes shapes and animation. We also discuss practical details to the integration of our methods into MPEG-21 and MPEG-4 architectures.

1. Introduction

1.1 Targeting a Context

Three-dimensional (3D) representation is one of the cornerstones of computer graphics and multimedia content. Advances in this domain, coupled with the highly fuelled progression of 3D graphics cards has pushed the complexity of these representations into a whole new arena whereby a single real-time model can consist of more than a million polygons. Huge architectural buildings, everyday objects, even humans themselves can be represented using 3D graphics in such detail it is difficult to distinguish between real and virtual.

Concurrently, and much towards the other end of the scale, many devices (Personal Digital Assistants (PDAs), Mobile Phones, Laptops, etc) are now “3D capable” to enhance a user’s experience and provide much more depth to the information presented. In many cases these devices access the same content from the same service provider (for example: providing virtual maps/guides, multi-user games etc) and it is this broadness of content and the heterogeneity of devices (in terms of performance, capability, network connection etc) that is the main concern in a continuously expanding market. It is also the concern of the user to obtain the best quality for their device, i.e. a general expectation of any device of higher performance is that overall the quality of the experience will be better.

1.2 Graphics DIA

DIA offers a solution to these current industry woes and whilst most of the more common examples are in the area of video and audio, the popularization of applications shows that there is also a popular trend for 3D graphics. Whilst there are many concerns for 3D graphics as a whole, as it is a very large domain already, we focus our attention in this paper solely on representation, and emphasize on the constraints that need to be considered in the domain of virtual humans. In addition, whilst the research area of multi-resolution models [1] is one that has been significantly explored we introduce it into the domain of DIA in order to consider the constraints imposed by multiple targets, and the adaptation from a single content.

Our main research goal was to devise a method for multi-resolution representation/adaptation to transfer and render a virtual human (face and body) on heterogeneous targets devices with the following considerations:

• **Range of Complexity** - As the model will not be targeted specifically for a particular network or device, we needed to define a generic method. In essence this means the ability to represent a virtual human from its lowest possible representation towards its most complex

• **Device Capability** - We defined a generic method for assessing the capability of the device to enable the closest approximation. As the target device is
not specified, it should be assumed that this assessment should also be heterogeneous

- **Network Capacity** - Based on an assessment of the network condition we used a generic method for adaptation that allows for both a specific bandwidth and maintains a certain level of Quality of Service

- **Interoperability** – It is a necessity that whilst the server is aware of an adaptation of the content towards a specific device, awareness for the client should not be required. Hence, regardless of whether the content is adapted or not, it should be useable on the target device without any specific indicators. The reasons are many, but the main one is that the decoding process should not require additional information as this would create (a) a huge dependency (b) updates to an already defined schema; in addition it is also not necessary as long as the process is thoroughly considered

- **Animatable Models** – The models themselves are not just representations, they must be also animated, which means we needed to consider ramifications at the joint (body) and control point level (face) of simplification

- **Simple Adaptation** - Adaptation should be simple enough to deal with any types of clients. When an adaptation process is done in a single server, it is quite crucial to cope with the wide range of devices

- **Compact Data Form** - The model should be compact in order to represent all different levels; this is essential because whilst the main priority of MPEG-21 is to allow for adaptation, it should not negate the objective of MPEG-4 for a compact form.

In this paper, we present a new representation and an adaptation method of multi-resolution model. We present in Section 2 our approach, including constraints imposed by the schema. Section 3 discusses the issues involving heterogeneous targets, and Section 4 presents the experimental results with MPEG schemes; with Section 5 concluding the paper.

2. Adaptation Overview

2.1 Overall Configuration

Although MPEG-21 DIA [9] is not restricted to a Client/Server (Figure 1) architecture it is currently the most commonly used setup for most applications; however some peer-to-peer type applications are emerging.

The basis of adaptation within MPEG-21 is quite complex on the outside due to its generic requirement and there are several elements that are required to perform a complete adaptation (illustrated in Figure 1); these are as follows:

- **Content Digital Item (CDI)** – The original encoded format of a CDI is required (although it can be stored in other elements, it is simpler if it is treated separately). This DI should be the highest quality version and contain all elements required.

- **Generic Bitstream Description (gBSD)** – The gBSD file, introduced by Amielh et al [5, 11], as it is commonly known is essentially the mapping of each of the “important” elements within the encoded file. It can be generated many ways, but the simplest is during the encoding of the original DI. It is basically an XML representation marking lengths, offsets and other information that enables the adaptation to occur.

- **XML StyleSheet (XSLT)** – The XML StyleSheet is basically the engine for the adaptation. This stylesheet relates the gBSD to the actual values or preferences given by the user or the terminal. It contains processes for verification of a gBSD file against the required adaptation process, as well as the adaptation process itself. This is generally an execution routine and must be handcrafted for a particular task or set of tasks.

- **Context Digital Item (XDI)** – The XDI contains the formatted information from the client (and can also contain information from the server) on the target device, the user preferences and is basically the input control parameters for adaptation.

The adaptation process is basically passing information through the engine and the adapted file (ACDI) is constructed from elements that have been edited (mainly in the header) and elements that have passed a specific criterion, but generally have remained unchanged. For example, and as illustrated in Figure 2, if the adaptation process indicates frames 5 to 20 should be removed the header should indicate this change (to maintain consistency – so that the decoder does not crash or become unstable). Frames 1 to 4 and 21+ should pass through the adaptation process without change, but 5 to 20 will be dropped and not passed onto the adapted file. Figure 3 illustrates this
2.2 Representation of Content

There have been a few approaches to adapt multiresolution techniques for the transmission of complex shapes including progressive meshes (PM) [2, 3, 4]. Adaptation within MPEG is a relatively new topic and as such only a few methods have been proposed [5] mainly dealing with audio/video media; whilst 3D graphics has received little attention [6]. Here we present our research on the development of an adaptation schema using both MPEG-4/21 schemas.

As the gBSD is used to define the bitstream layout on a high level it must basically represent a decoder in XML format. This does not mean that it will decode the bitstream, but the structure of the bitstream is important. This means that the adaptation methods must be inline with the lowest level defined in the schema. For example, if the adaptation needs to skip frames, and as MPEG codecs are byte aligned, it is practical in this case to set a marker at the beginning of each frame; this means that a frame can be dropped without the need to understand most of the payload (this is possibly with the exception of updating the frames-skipped section of the header – as this could be avoided). However, as will be seen in the following sections the schema is based on a much lower level in order to be more flexible.

3. Content Preparation

3.1 Multi-resolution Model Generation

We devised an efficient method of adaptation using a clustered hierarchical model. The premise involves clustering all the data so that a specific complexity can be obtained by simply choosing a set of clusters. From the complex mesh $M_a(V_n,F_n)$ where $V_n$ is a set of vertices and $F_n$ is a set of faces, it is sequentially simplified to $M_{a_1},...M_{a_l}$. A multi-resolution model of this simplification sequence has, or at least is able to generate, a set of vertices $V$ and faces $M$, where union is denoted as ‘$+$’ and intersection is denoted as ‘$-$’:

$$V = \sum_{i=0}^{+} V_i, M = \sum_{i=0}^{-} M_i$$ (1)

$V$ and $M$ can be partitioned into set of clusters. The first type is a set of vertices and faces that are removed from a mesh of the level $i$ to make a mesh of the level $i-1$, denoted by $C(i)$. The other type is a set of vertices and faces that are newly generated by simplification, denoted by $N(i)$. Hence a level $i$ mesh is as follows:

$$M_i = M_0 + (\sum_{j=1}^C C(j) - \sum_{j=1}^N N(j))$$ (2)

There are many simplification operators, including decimation, region merging, and subdivision [1]; here we used half edge-collapsing operators [2] and Quadric Error Metrics [6].

By an edge-collapsing operator, an edge $(v_1,v_2)$ is collapsed to the vertex $v_0$. From the example (Figure 4), faces $f_1,f_2$ are removed from the mesh, and faces $f_3,f_4$ are modified into $f'_3,f'_4$. The clusters are defined as:

$$C(i) = \{f_1,f_2,f_3,f_4\}$$
$$N(i) = \{f'_3,f'_4\}$$

To evaluate the equation (2), it requires set union and intersection, which are still complex. By the properties of the simplification, it ensures $N(i)$ to be a subset of unions of $M_0,C(1),...,C(i-1)$. Using this property, the cluster $C(i)$ is sub-clustered into a set of $C(i,j)$, which belongs to $N(j)$ where $j<i$ and $C(i,j)$ which does not belongs to any $N(j)$. It is same for the $M_0$ where $M_0=C(0)$. Thus, the level $i$ mesh is represented as the following equation, which requires simple set selections.

$$M_i = \sum_{k=0}^{C(k,0)} (C(k,k) + \sum_{j=1}^{N} C(k,j))$$ (3)
The last process is the ordering, to reduce the number of selection during the adaptation process. Ordering the vertices data is rather straightforward, because the edge-collapsing operator \((v_i, v_s)\) ensures that every \(C(i)\) has a single vertex \(v_i\) as \(C(i, i)\). By ordering vertices of \(C(i, i)\) by the order of \(i\), the adaptation process of vertex data for level \(i\) is a single selection of continuous block of data, \(v_0, v_1, \ldots, v_i\). For the indexed face set, each \(C(i)\) is ordered by \(C(i, j)\) in the ascending order of \(j\). Thus, an adaptation to level \(i\), consists of at most \(3i+1\) selections or at most \(2n\) removals.

So far, we have described the process using only the vertex positions and face information. In the mesh, there are other properties that have to be taken into account, such as normal, color, and texture coordinates. Because these properties inheritably belong to vertices, a similar process to vertex positions is applied. Exceptional cases are; 1) two or more vertices use the same value for a property, 2) a single vertex has more than two values. In both cases, there is a unique mapping from a pair of vertex and face to a value of properties. The cluster \(C(i)\) has properties which have mapping from \((v_i, f_j)\) where \(v_i \in C(i)\). If a property \(p\) belongs to more than one vertex such as \((v_i, f_1) \rightarrow p\) and \((v_j, f_2) \rightarrow p\), \(p\) is assigned to the cluster of \(C(j)\) where \(j < i\). By ordering this, \(p\) remains active as long as there is one vertex that has \(p\) as its property. Therefore, we have a valid set of clusters for each level \(i\).

Figure 5 shows the idea of clustering. Each cluster has a set of vertices and vertex properties such as vertex normal, colors, and texture coordinates. Along with vertex information, the cluster has a set of indexed faces, normal faces, color faces and texture faces. Also each cluster can be consists of sub-segments with its own material and texture. Each level is selected by choosing blocks of clusters.

In real applications, it is often not necessary to generate all levels of details. Differences by a few polygons do not usually give significant differences either in performances or in qualities. By using the proposed representation, the modeling system is able to generate a set of levels with any combination depending on a model and application.

### 3.2 Animation Data Construction

After the adaptation, the animation data is applied to adapted geometry. For the facial animation, we use the MPEG-4 Facial Animation Parameters (FAP) to drive our facial animation engine. These parameters provide information about the displacement of FAP control points, to change the face shape during animation. However, the FAP stream does not provide any information for the displacement neighboring vertices.

From the FAP information, we define each displacement, i.e. which vertices are influenced and in which direction according to FAP intensity. If the control points have a large range of movement such as lip parts, the FAP intensity would be large. We define two approaches in order to animate the face model according to FAP information.

Firstly, we could use a piece-wise linear interpolation of FAP control points during animation in order to animate the face model. This technique requires low computational cost during animation but the generation of influence region at the modeling...
stage is not simple. A designer could define this information, but due to a lot of overlapping areas of a FAPs influence (especially around the lips) makes this work difficult. In addition, this needs to be done for each model and for each level of detail. This technique could not be applied to any automated process.

Another technique would be to use geometric deformation algorithms to compute a FAP’s influence. This technique allows an automatic computation of influenced vertices according to Facial Definition Parameters (FDP), the position of each Feature Control Point, and the mesh topology. In few seconds, the facial animation engine is able to animate a face model with only this information. However, this is not feasible on devices with low computational power as it is time consuming and no possible for real-time animation.

The best solution that can be applied to a wider range of platforms is to automatically export piecewise linear interpolation information from a geometric deformation engine and use it during animation. From the high-level resolution model, we automatically construct the Facial Animation Table, which describes influences of FAP points and information for the interpolation. As the vertices of the model were ordered by FDP and influence, we can easily extract the corresponding FAT information from a high-level FAT table, for each model; therefore only the corresponding part of the global FAT table is then transmitted to the client.

4. MPEG-21 Adaptation

4.1 Overview

Whilst MPEG-21 is based on the principle of adaptation from peer-to-peer, it is currently based on the concept of Server/Client type, whereby content is adapted at the Server side and streamed to a client.

For the adaptation, we focus on the entire context of adaptation instead of a particular context of network capacity or performance of a rendering hardware. As previously described there are many different target contexts to which the mesh can be adapted; the main consideration is to make sure that the mesh is adapted to satisfy all contexts. As an overview, the network, the device capability and the user’s preferences/restrictions are the main considerations. In the context of the network, we are able to use bandwidth measurements / figures in order to determine the available capacity. The user’s preferences generally consist of choosing the LoD explicitly (if the application allows). The terminal’s capability is the most complex, and is dependant on many factors; in this case we use device benchmarking.

4.2 Device Benchmarking

In order to solve one of the essential elements of DIA (i.e. concisely describing the terminal’s computational power) we introduced into the schema the concept of benchmarking for both Digital Item, and Terminal. This was not only for graphics, but the principle was generic and can be used for video and audio media types as well.

As part of our initial investigation, we use a linear approximation that is attributed to any digital item. In essence, and in terms of graphics, we use the following procedure in order to approximate the level of complexity that should be adapted to.

- **Digital Item Benchmark** – The digital item is executed on a standard test device, at its highest resolution. The test device is expected to enable the decoding and rendering of the digital item, at a suitable frame rate, which provides the value $F_N$ (for video and graphics, greater than 25fps, in terms of audio at the desired sampling rate).

- **Test Device Benchmark** – The test device is benchmarked under normal test conditions; this involves reducing the number of concurrently running applications to a minimum. The most appropriate benchmark is used, more than one if possible; this provides us with a value $B_T$.

- **Target Device Benchmark** – As part of the runtime execution, the target device is benchmarked (either prior to runtime or as part of the initialization process) and provides us with a value $B_N$. $B_T$ and $B_N$ should be of the same type; either for a specific matching benchmark, or a weighted sum (again with corresponding elements).

- **Adaptation Ratio** – If the target device is required to run at a value of $F_T$, the mesh should be adapted by using a ratio $R_A$ (in our case we directly reduce the number polygons by this factor, using the closest approximation). In all cases, we need to account for the test machines excess and the target machines possible limitations; we therefore use Equation (4) in order to compute the mesh reduction ratio.

$$R_A = \frac{F_N}{F_T} \times \frac{B_T}{B_N} \times C_R$$  \hspace{1cm} (4)$$

In this case, for all cases where $R_A$ is less than 1.0 adaptation is required, for all other cases the mesh representation remains unchanged. $C_R$ is an additional factor which can remain at 1.0, but can be used to allow the user to specify how much computational power should be taken by the processing of a specific digital item.
An additional factor is that many digital items might be decoded and rendered on the same device at the same time (a graphics animation and audio stream for example). However, using this method, it is simple to account for two or more digital items on the same machine, as the value $C_R$ can also be used for the ratios attributed to each digital item as long as the sum of all $C_R$ values is equal to 1.0.

Our method is very effective because it permits adaptation without the need to (a) comprehend the various factors within computer (e.g. CPU type, Frequency etc. which requires high assumption on the hardware architecture) and (b) reduces the process to a simple linear approximation.

4.3 Network Capacity

The adaptation of a mesh in terms of the network capacity is directly governed by the available bandwidth $C_B$, the encoded file size $F_S$ (as the mesh is mainly adapted after it has been encoded), and the download time $T_W$ (as a mesh is not dynamic) that a user is prepared to wait or that a provider believes reasonable; exemplified in Equation (5).

$$\frac{C_B}{F_S} \times T = R_A$$ (5)

$R_A$ provides the ratio of the original file to the adapted file, and as the main elements contributing to this size (assuming that the encoding process is well balanced) is the mesh size, then the size of the mesh is reduced. Values for $F_S$, $C_B$, and even $T_W$ are easily obtained and therefore the value for $R_A$ is also easily determined.

4.4 Additional Considerations

The adaptation process does not require additional memory other than storage for clusters to be processed. The adaptation process does not require additional memory other than storage for cluster to be processed and adapted mesh data which makes the process applicable to low-end machines with less memory. The client-side adaptation process generates a set of clusters with lower levels of details that can be dynamically adaptable during the rendering time, in a very similar way as Level of Details works on standard desktop PCs.

For facial animation, after the adaptation, the model has the corresponding Facial Animation Table. During animation, a real-time deformation engine compile each FAT according to the current set of FAP, and compute deformation of the mesh by merging each FAT. It requires little computational cost each frame by simplifying the FAT generation process as a simple selection of sub-sets. This technique works well on powerful platform but is also appropriate for platform with lower computation like a PDA device.

5. Experimental Results

Table 1 shows the benchmark results for different target devices. As a basis for graphics we used the
ViewPerf benchmark for SPEC [9] and compared the values with actual performance of an actual application, called VHD++ [12], with virtual buildings and character animation. The benchmark result of each ViewPerf test set is averaged by geometric sum [9]. Figure 8 illustrates that the averaged benchmark result approximates to the actual performance of the real application (with the slight, although within acceptable limits, exception of Application Test # 2, on machine 3). By utilizing different benchmark results for different data sets and applications, we can approximate the performance more closely. For this case, Application # 2 could utilize a subset of benchmark such as light-06 and 3dsmax-02, which will describe its performance more closely.

In order to verify our proposed method we applied a set of human body models with both body and face animation. Multi-resolution models were generated based on the benchmark for that device and the resulting size values tabulated in Table 2 and illustrated in Figure 10.

The progressive mesh approach is known as a method using near an optimal storage for multi-resolution approach. The discrete mesh is a representation of a set of discrete levels of details which is still quite common in most of real-world applications because of its simplicity in adaptation.

The numbers are the size of the VRML and BIFS files respectively. Since the PM is not able to be encoded to BIFS format, only the approximated size for the VRML file is noted. As a result of the adaptation, the highest details have a number of polygons of 71K and 7K for each model whilst the lowest details have 1K and 552 polygons each. The models are constructed to have 5 different levels. The body model has 12 operation of adaptation for each segment, which has vertex normal and texture coordinates as properties. It is quite small when compared to operations of PM, which requires at least n/2 operations, where n is number of vertices (28K for the body model). Furthermore, the operation is simple selections whilst PM requires relatively complex substitutions.

The proposed method is located in-between of these approaches, and is flexible and simple enough to allow adaptation with relatively small file size. Although the proposed method has larger data to the PM approach, it is encodable to standard codec and is able to be transmittable via standard MPEG streams. It also utilizes a simpler adaptation mechanism, which is very similar to the simplest discrete level selection.

### 6. Conclusion

In this paper we have presented a method to format 3D representation data that can then be encoded and then adapted using a generic schema within the MPEG-21 framework; essentially the representation is produced in a way that requires no additional information at the client side and can be rendered immediately. In addition, the representation used is suitable for clients with low computational power, and provides significant advantages over progressive mesh techniques of devices such as PDAs. We have also introduced a benchmarking method that allows a linear approximation to be used for the adaptation process of the mesh representation, reducing the overall polygon density to match the capabilities of the device. Whilst the method presented only approximates the adaptation of a representations polygon density to a specific value.
this approximation is well within acceptable limits and it is not intended, nor could it be hoped, to be more precise; there are just too many mitigating factors to make it possible.

7. Future Work

The context within MPEG-21 Digital Item Adaptation for adaptation is very generic, and in many respects does not even need to apply to MPEG encoding/representation schemes; therefore we are currently determining if there is a better approximation for the benchmarked result, and the polygon mesh adaptation, possibly based on a logarithmic schema. We are also looking to link the result more closely with other sets of terminal descriptors such as display capabilities (e.g. defining the screen resolution etc), and the memory capacity.

We are also concerned with the dynamic adaptation of both representation and animation in harmony with each other. This mainly involves transmitting a representation based on the Level of Articulation (LoA) used, which greatly reduces the overall polygon count; whilst at the same time maintaining the possibility for a dynamic increase in LoA. In addition we are researching methods for multi-resolution textures suited for similar applications.

Acknowledgements

The research presented has been funded through the European Project ISIS (IST-2001-34545) by the Swiss Federal Office for Education and Science (OFES). The authors would also like to thank Lionel Egger for his valuable test models and consultation.

References


Figure 10. Examples of Adapted Meshes
(Top from left: body models with 71K, 45K, 30K, 15K and 5K polygons)
(Bottom from left: face models with 7K, 5K, 3K, 2K and 0.5K polygons)