DOWNLOAD PDF LEVEL OF DETAIL FOR 3D GRAPHICS

Chapter 1 : Level of Detail in 3D graphics - What are the pros and cons? - Stack Overflow

Level of detail or LOD is a discipline of interactive computer graphics that attempts to bridge complexity and performance by regulating the amount of detail used to represent the virtual world. Level of detail is as relevant today as ever, for despite tremendous strides in graphics hardware.

Even offline rendering of animation and visual effects, which does not require interactive rates, can benefit from regulating level of detail. The fundamental concept of LOD, summed up in Figure 1. This less detailed representation typically consists of a selection of several versions of objects in the scene, each version less detailed and 6 Chapter 1 Introduction faster to render than the one before. Generating and rendering these progressively coarser versions of objects, known themselves as levels of detail or LODs, has become an extremely hot topic over the last decade. Dozens of LOD algorithms have been published, and dozens more have undoubtedly been whipped up by developers unaware of or confused by the menagerie of techniques available. This book presents a detailed treatment of the field of LOD, distilling into a single volume research, theory, and algorithms from journals and conferences spanning several countries and several years. The book also aims to provide an applied resource that will help developers integrate state-of-the-art LOD into their games, simulations, and visualizations. In our field an unfortunate gulf separates much research from implementation; many excellent algorithms have never made it from the ivory towers of academia to the practical world of industrial development. This book aims to bridge that gulf. Generation, Application, and Advanced Issues. Part I discusses frameworks and algorithms for generating levels of detail. We focus here on the most common task in LOD: Once an entirely manual process, a great many algorithms have been proposed in the last decade to perform this simplification. In Chapter 2 we categorize and discuss work on mesh simplification, emphasizing the underlying similarities and differences among algorithms: We also discuss issues of mesh topology, since modifying that topology e. Finally, in Chapter 3 we focus on error metrics in simplification. We examine not only geometric error but also analytic and image-based measures of error for surface attributes, such as colors, normals, and texture coordinates. Part II focuses on the task of managing the level of detail used to render a scene. Here the tradeoff between fidelity and performance becomes concrete: We move from the general to the specific, opening with a discussion of run-time frameworks for level of detail in Chapter 4. Such frameworks may assemble the scene from LODs created ahead of time, or they may generate LODs on the fly, tailoring them to the particular viewing scenario. Chapter 5 provides a catalog of useful algorithms related to LOD, including several approaches to mesh simplification. We have chosen algorithms that span the gamut of simplification 1. Chapter 6 addresses special topics of particular applied interest: The most mainstream application of computer graphics, video games impose unique constraints, such as tight polygon and computation budgets, as well as unique opportunities, such as a staff of full-time artists to manually create or optimize LODs. Chapter 7 is devoted to the special case of terrain simplification. Terrains and height fields have a specific structure and often tremendous complexity that enable and require specialized algorithms. Many innovations in mesh simplification were first invented for the particular case of terrains. Part III turns to more theoretical, but no less important, topics. These are the deeper questions underlying level of detail management. What principles of visual perception can help us design better LOD algorithms? How can we evaluate the perceptual error introduced by our simplifications? What is an appropriate frame rate, and how does it vary by task? In Chapter 8, we begin with an overview of human visual perception, present a simple model of lowlevel vision suitable for driving LOD, and describe some algorithms based on variations of that model. In Chapter 9, we examine techniques to evaluate visual fidelity in level of detail. We describe automatic and experimental measures of visual fidelity, and discuss the implications of recent research comparing these measures. Finally, in Chapter 10, we consider temporal issues. How do the effects of latency, frame rate, and overall system responsiveness differ? Is a fixed frame rate always best? Recognizing the redundancy of using many polygons to render an object covering only a few pixels, Clark described a hierarchical scene graph structure that incorporated not only LOD but other now common techniques, such as view-frustum culling. We highly recommend this paper to readers interested in the historical genesis of

computer graphics. Flight simulators were probably the earliest applications to make extensive use of LOD [Cosman 81]. In these early days, LODs were created by handâ€”considering the 8 Chapter 1 Introduction overall cost of a flight simulator system, the cost of paying a designer or artist to create multiple models of each object was insignificant. The early s saw a sudden interest in automating this process, and researchers published a flurry of papers describing algorithms for simplifying a highly detailed model while still capturing its visual appearance. Some of those first algorithms, such as vertex decimation [Schroeder 92] and gridded vertex clustering [Rossignac 92], remain among the most important and useful; we describe them in detail in Chapter 5. Over the years other algorithms and frameworks for LOD have been developed. The many key developments, which we will elucidate throughout the book, include optimization-based predictive schedulers for selecting LODs [Funkhouser 93b], progressive meshes for continuous LOD [Hoppe 96], vertex hierarchies for view-dependent LOD [Hoppe 97][Luebke 97][Xia 96], quadric error metrics for measuring simplification error [Garland 97], guaranteed bounds on surface and texture distortion [Cohen 96][Cohen 98a], and principled simplification of topology [He 96][El-Sana 99b]. Today the graphics developer faces a bewildering array of choices. Simplification algorithms range from simple and fast to sophisticated and slow, and the resulting LODs range from crude to excellent. This book should provide a guide to the many techniques for creating and managing LODs, as well as an understanding of the deeper underlying principles. Some simplifications are so woven into the practice of computer graphics that we tend to forget about them, such as the quantization of color to 24 bits or the use of three-color channels red, green, and blue to represent the spectral response of the virtual scene. Even storing a polygonal mesh may involve simplification, since we clamp the precision of the coordinates and normals typically to 32 bits each for X, Y, and Z. Although useful and long-used approximations, especially in interactive graphics, the quantization of color and geometric attributes is indeed simplification, and recognizing this can lead to many benefits. In general the line between simplification and lossy compression techniques, which include quantization in its various forms, can be blurry. In the main, however, we will define LOD and simplification throughout this book as processes that reduce the complexity of polygonal meshes, not their precision or storage size. Shading and illumination form another spectrum of simplification in computer graphics. Many packages let the user control rendering performance with the familiar progression from wireframe to flat shading, to Gouraud shading, to texture mapping, and finally to per-pixel effects, such as Phong shading, bump mapping, and 1. Lighting calculations range from simple local illumination techniques, such as the Phong model still used for the majority of interactive graphics, since this is what most hardware supports , to sophisticated global illumination methods such as radiosity and path tracing. Designers have long recognized the possibility of managing rendering performance by varying the complexity of shading and illumination applied to objects within the scene. The connection between texture mapping and level of detail deserves particular mention. For example, Dumont et al. A more frequent technique, however, replaces geometric detail in LODs with texture maps. Termed imposters by Maciel and Shirley [Maciel 95], these textured LODs are especially common in video games and visual simulation applications. Although we chiefly focus on geometric level of detail management, we discuss the use of textured imposters and backdrops in Chapter 6. We begin with an overview of three basic frameworks for managing level of detail: The original scheme proposed by Clark in and used without modification in most 3D graphics applications today, this approach creates multiple versions of every object, each at a different level of detail, during an offline preprocess. At run-time the appropriate level of detail, or LOD, is chosen to represent the object. Since distant objects use coarser LODs, the total number of polygons is reduced and rendering speed increased. Because LODs are computed offline during preprocessing, the simplification process cannot predict from what direction the object will be viewed. The simplification therefore typically reduces detail uniformly across the object, and for this reason we sometimes refer to discrete LOD as isotropic or view-independent LOD. Discrete LOD has many advantages. Decoupling simplification and rendering makes this the simplest model to program: Furthermore, modern graphics hardware lends itself to the multiple model versions created by static level of detail, because individual LODs can be compiled during preprocessing to an optimal rendering format. For example, depending on the particular hardware targeted, developers 10 Chapter 1 Introduction may convert models to use features such as

triangle strips, display lists, and vertex arrays. These will usually render much faster than simply rendering the LODs as an unordered list of polygons. Rather than creating individual LODs during the preprocessing stage, the simplification system creates a data structure encoding a continuous spectrum of detail. The desired level of detail is then extracted from this structure at run-time. A major advantage of this approach is better granularity: This frees up more polygons for rendering other objects, which in turn use only as many polygons as needed for the desired level of detail, freeing up more polygons for other objects, and so on. Better granularity thus leads to better use of resources and higher overall fidelity for a given polygon count. Continuous LOD also supports streaming of polygonal models, in which a simple base model is followed by a stream of refinements to be integrated dynamically. When large models must be loaded from disk or over a network, continuous LOD thus provides progressive rendering and interruptible loadingâ€”often very useful properties. Thus view-dependent LOD is anisotropic: For instance, nearby portions of the object may be shown at higher resolution than distant portions, or silhouette regions of the object shown at higher resolution than interior regions Figures 1. This leads to still better granularity: This in turn leads to still better fidelity for a given polygon count, optimizing the distribution of this scarce resource. Indeed, very complex models representing physically large objects, such as terrains, often cannot be adequately simplified without view-dependent techniques. Creating discrete LODs does not help: The field of view is shown by the two lines. The model is displayed at full resolution near the viewpoint and at drastic simplification far away [Luebke 01a]. Unsegmented or poorly segmented data, such as the skeleton extracted from an MRI scan shown in Figure 1. Here almost the entire data set forms one connected surface, preventing the use of discrete LODs to speed rendering and preserving detail in the area of interest. Another source of difficult models is scientific visualization, which tends to produce extremely large data sets that are rarely organized into conveniently sized objects. Again, view-dependent LOD can enable interactive rendering without manual intervention or extra processing for segmentation. Note that the polygons facing away from the viewer, shown here for clarity, would typically be culled away by backface culling. View-dependent LOD, despite its advantages, also comes with some significant drawbacks. The extra processing required for evaluating, simplifying, and refining the model at run-time, and the extra memory required by the view-dependent data structures, steer many developers away from this approach. Video games, often the most demanding and tightly optimized of graphics applications, have been notably reluctant to adopt view-dependent LOD outside of situations that almost require it, such as terrain rendering. Continuous LOD also imposes a cost in processing and memory, but has fared somewhat better in game usage.

Chapter 2 : Redefining the Level of Detail for 3D Models

This website provides resources supporting the book Level of Detail for 3D Graphics by D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner; published by Morgan Kaufmann as part of their series in Computer Graphics and Geometric Modeling.

Level of Detail Introduction 3D graphics has become an indispensable part of the computer users landscape. Applications such as CAD, games and scientific visualisation allow the user to navigate, manipulate and interact with an approximation of a 3D world, usually through a two dimensional display. The demand of users and the goal of computer graphics researchers is to improve the accuracy and realism of this approximation. In order to store, process and render 3D objects and scenes using computer graphics, contemporary real-time interactive 3D graphics systems utilize polygon meshes. A polygon mesh is a collection of vertices and polygons that define the shape of an object. Polygon meshes are flexible, easy to understand, allow for the encoding of detailed surface features and are naturally amenable to processing by computational methods. Other representations are available, for example constructive solid geometry CSG and volumetric grids, these are used where interaction speed is not a priority or are used as an intermediate representation before being converted into a mesh. Real world scenes can be arbitrarily complicated. Interactive system need a fast frame rate to provide an satisfactory user experience. Graphics hardware is limited in the number of primitives e. Computing for lighting and global effects further reduces the number of primitives which can be processed. Therefore implementers of graphics systems are faced with a tradeoff between rendering speed and scene complexity or scene realism. A more sophisticated approach introduced by James Clarke recognized the redundancy of using many polygons to render an object covering only a few pixels. Clarke described a scheme where each object had a number of different representations Levels of Detail. The representation used to render the object was dependant on the expected screen space occupied by the object. Essentially, as an object moves away from the viewer in perspective projection system, increasingly simpler mesh coarser approximations can be used to render the object. Early practitioners created LODs by hand, but this becomes impractical as models became larger and more complex. The advent of range scanners and medical imaging devices which can scan convert real objects in to multi-million polygon meshes created a need for automated level of detail creation. An automated LOD procedure should be able to significantly reduce the number of polygons while maintaining a good approximation to its visual appearance. Approaches to level of detail management Application of LOD over a wide variety of type of graphics applications has resulted in the development a number of approaches to LOD management. A number of versions of an objects are created, each with a different level of detail during an offline process. At runtime an appropriate version is used to render the object. The choice of version used by the rendering engine determined by estimated screen space error or the need to reduce scene complexity to maintain frame rate or usually a combination of these two factors. Discrete LODs are produced offline, therefore have the advantage of having no constraint on time required to generate the simplifications. The simplification can be further optimised to take advantage of certain features within graphical hardware. For these reasons Discrete LOD is the most widely used approach. As processing is done offline, discrete LOD can not predict from what direction the object will be viewed, so the simplification process reduces detail uniformly across the model. In this case relatively large objects which have a large spatial range in the direction away from the viewer may have redundant detail in the parts of the model that are relatively distant. An additional problem is the rendering of silhouettes or the outline region of a projected object. The human eye visual system is very good in spotting discontinuities straight edges and angles in ostensibly continuous boundary. Continuous LOD A data structure encoding a continuous spectrum of detail is created offline. The key advantage of this approach is that the exact level of detail required for a model in a given situation can be quickly extracted from the data structure. The allows the engine to use only the exact number of polygons needed, freeing up polygons for other parts of the scene. Areas of a single object can be rendered with different level of detail. Regions where visual fidelity is important areas close to the viewer, silhouettes, perceptually important areas, etc. Terrain

rendering is a field which benefits greatly. This approach gives is an ideal solution where the polygon budget is a priority, as polygons can be used where most needed. The major drawback to View Dependant LOD is the fact that simplification needs to be done in real-time. In most situations the processing overhead required usually outweighs the gains in polygon efficiency. Measuring Error In order to inform the application of simplification operators see below , a measurement must be made of the estimated error caused by applying the operator. Knowledge of the error is also used by LOD management systems to determine which Model to render in a give situation. Geometric error Most systems use geometric based metrics. Geometric measure are concerned with maximum and average distance between corresponding vertices, vertices and closest surfaces or inter-surface distance. These type of metrics are particularly useful when deciding how to apply a local simplification operator. Image error Geometric error is an indirect surrogate for the error to be minimised, namely, the errors in the rendered image. In this scheme, the impact of simplification operators on the image fidelity was measured and the operator with the smallest effect on the image applied. This approach is very slow when compared to geometric algorithms, there are also potential problems caused by undersampling the visual space around the model. Finally, the LODs produced are dependant on rendering parameters lights, shading model etc and may be sub optimal for different rendering conditions. Mesh simplification Local Simplification techniques The goal of mesh simplification is to reduce the complexity of the mesh number of polygons while maintaining the maximum visual fidelity. A large number of local simplification operations have been described in the literature. An operator can be applied to any part of the mesh. The choice of where and how to apply an operator is informed by measuring the error e between the original and simplified mesh. The typical approach is to apply the operator which minimises e. The measurement, more accurately, the estimation of e is a difficult problem in itself and will be discussed later. Repeated application of local simplification operators will produce a mesh of desired complexity. Local Simplification Operators Edge Collapse Edge Collapse removes an edge and its two vertices and two triangles from the mesh. A new vertex is created, then edges in the neighborhood edges which share endpoints of the removed edge are connected to the new edge. Edge collapse can cause mesh foldovers. Mesh foldover can cause undesirable effects such as discontinuities in texturing and lighting. A mesh foldover is detected by comparing the normals of the triangles before and after the edge removal. Vertex Pair Collapse Vertex Pair collapse amalgamates two unconnected vertices. The effect is similar to the collapse of an imaginary edge joining the two points, This is useful for filling holes in the mesh. As the number of unconnected pairs of vertices in a mesh is quite large, O n2 , heuristics have been developed to reduce the number of candidates. Triangle collapse A Triangle collapse removes a triangle, replacing it with a new vertex. The neighborhood of the new vertex will be neighborhood of the triangles original 3 vertices. Cell Collapse Cell Collapse collapses all vertices in a given area to one vertex. The area can be based on a regular grid or a volume surrounding vertices. Vertex removal Vertex removal removes a vertex, its edges and triangles, then re-triangulates the resulting hole. Polygon Merging Polygon Merging removes a number of adjacent almost coplanar faces and re-triangulates the hole. Global simplification Global Simplification operators consider the entire topology of the object fine tuning the parameters of a global operator will produce a mesh within specified error bounds. Techniques here include Volumetric processing and simplification envelopes. Level of Detail- Gaming Optimizations.

Chapter 3 : Level of Detail for 3D Graphics - PDF Free Download

Level of Detail for 3D Graphics (02) by Luebke, David - Reddy, Martin - Cohen, Jonathan D - Varshney, [Hardcover ()] LOD - Level of Detail - is a very important subject in computer graphics.

More info See in Glossary in the sceneA Scene contains the environments and menus of your game. Think of each unique Scene file as a unique level. In each Scene, you place your environments, obstacles, and decorations, essentially designing and building your game in pieces. More info See in Glossary is a long way from the cameraA component which creates an image of a particular viewpoint in your scene. The output is either drawn to the screen or captured as a texture. More info See in Glossary , the amount of detail that can be seen on it is greatly reduced. However, the same number of triangles will be used to render the object, even though the detail will not be noticed. By default, the main camera in Unity renders its view to the screen. More info See in Glossary allows you to reduce the number of triangles rendered for an object as its distance from the camera increases. It is built on top of the lower level transport real-time communication layer, and handles many of the common tasks that are required for multiplayer games. More info See in Glossary will reduce the load on the hardware and improve rendering performance. A GameObject can contain any number of components. Unity has many built-in components, and you can create your own by writing scripts that inherit from MonoBehaviour. Full details are given on the component reference page but the images below show how the LOD level used to render an object changes with its distance from camera. The first shows LOD level 0 the most detailed. Note the large number of small triangles in the meshThe main graphics primitive of Unity. Meshes make up a large part of your 3D worlds. Unity supports triangulated or Quadrangulated polygon meshes. Nurbs, Nurms, Subdiv surfaces must be converted to polygons. More info See in Glossary: Note that the mesh has been reduced in detail smaller number of larger triangles: More info See in Glossary , for as many LOD levels as you like, a LOD group for the object with appropriate settings will be created for you automatically on import. The numbering convention assumes that LOD 0 is the most detailed model and increasing numbers correspond to decreasing detail. Did you find this page useful? Please give it a rating: Thanks for rating this page!

Chapter 4 : Level of Detail for 3D Graphics - O'Reilly Media

Level of Detail for 3D Graphics brings together, for the first time, the mechanisms, principles, practices, and theory needed by every graphics developer seeking to apply LOD methods. Continuing advances in level of detail management have brought this powerful technology to the forefront of 3D graphics optimization research.

Even offline rendering of animation and visual effects, which does not require interactive rates, can benefit from regulating level of detail. The fundamental concept of LOD, summed up in Figure 1. This less detailed representation typically consists of a selection of several versions of objects in the scene, each version less detailed and Team LRN 6 Chapter 1 Introduction faster to render than the one before. Generating and rendering these progressively coarser versions of objects, known themselves as levels of detail or LODs, has become an extremely hot topic over the last decade. Dozens of LOD algorithms have been published, and dozens more have undoubtedly been whipped up by developers unaware of or confused by the menagerie of techniques available. This book presents a detailed treatment of the field of LOD, distilling into a single volume research, theory, and algorithms from journals and conferences spanning several countries and several years. The book also aims to provide an applied resource that will help developers integrate state-of-the-art LOD into their games, simulations, and visualizations. In our field an unfortunate gulf separates much research from implementation; many excellent algorithms have never made it from the ivory towers of academia to the practical world of industrial development. This book aims to bridge that gulf. Generation, Application, and Advanced Issues. Part I discusses frameworks and algorithms for generating levels of detail. We focus here on the most common task in LOD: Once an entirely manual process, a great many algorithms have been proposed in the last decade to perform this simplification. In Chapter 2 we categorize and discuss work on mesh simplification, emphasizing the underlying similarities and differences among algorithms: We also discuss issues of mesh topology, since modifying that topology e. Finally, in Chapter 3 we focus on error metrics in simplification. We examine not only geometric error but also analytic and image-based measures of error for surface attributes, such as colors, normals, and texture coordinates. Part II focuses on the task of managing the level of detail used to render a scene. Here the tradeoff between fidelity and performance becomes concrete: We move from the general to the specific, opening with a discussion of run-time frameworks for level of detail in Chapter 4. Such frameworks may assemble the scene from LODs created ahead of time, or they may generate LODs on the fly, tailoring them to the particular viewing scenario. Chapter 5 provides a catalog of useful algorithms related to LOD, including several approaches to mesh simplification. We have chosen algorithms that span the gamut of simplification Team LRN 1. Chapter 6 addresses special topics of particular applied interest: The most mainstream application of computer graphics, video games impose unique constraints, such as tight polygon and computation budgets, as well as unique opportunities, such as a staff of full-time artists to manually create or optimize LODs. Chapter 7 is devoted to the special case of terrain simplification. Terrains and height fields have a specific structure and often tremendous complexity that enable and require specialized algorithms. Many innovations in mesh simplification were first invented for the particular case of terrains. Part III turns to more theoretical, but no less important, topics. These are the deeper questions underlying level of detail management. What principles of visual perception can help us design better LOD algorithms? How can we evaluate the perceptual error introduced by our simplifications? What is an appropriate frame rate, and how does it vary by task? In Chapter 8, we begin with an overview of human visual perception, present a simple model of lowlevel vision suitable for driving LOD, and describe some algorithms based on variations of that model. In Chapter 9, we examine techniques to evaluate visual fidelity in level of detail. We describe automatic and experimental measures of visual fidelity, and discuss the implications of recent research comparing these measures. Finally, in Chapter 10, we consider temporal issues. How do the effects of latency, frame rate, and overall system responsiveness differ? Is a fixed frame rate always best? Recognizing the redundancy of using many polygons to render an object covering only a few pixels, Clark described a hierarchical scene graph structure that incorporated not only LOD but other now common techniques, such as view-frustum culling. We highly recommend this paper

to readers interested in the historical genesis of computer graphics. Flight simulators were probably the earliest applications to make extensive use of LOD [Cosman 81]. In these early days, LODs were created by handâ€”considering the Team LRN 8 Chapter 1 Introduction overall cost of a flight simulator system, the cost of paying a designer or artist to create multiple models of each object was insignificant. The early s saw a sudden interest in automating this process, and researchers published a flurry of papers describing algorithms for simplifying a highly detailed model while still capturing its visual appearance. Some of those first algorithms, such as vertex decimation [Schroeder 92] and gridded vertex clustering [Rossignac 92], remain among the most important and useful; we describe them in detail in Chapter 5. Over the years other algorithms and frameworks for LOD have been developed. The many key developments, which we will elucidate throughout the book, include optimization-based predictive schedulers for selecting LODs [Funkhouser 93b], progressive meshes for continuous LOD [Hoppe 96], vertex hierarchies for view-dependent LOD [Hoppe 97][Luebke 97][Xia 96], quadric error metrics for measuring simplification error [Garland 97], guaranteed bounds on surface and texture distortion [Cohen 96][Cohen 98a], and principled simplification of topology [He 96][El-Sana 99b]. Today the graphics developer faces a bewildering array of choices. Simplification algorithms range from simple and fast to sophisticated and slow, and the resulting LODs range from crude to excellent. This book should provide a guide to the many techniques for creating and managing LODs, as well as an understanding of the deeper underlying principles. Some simplifications are so woven into the practice of computer graphics that we tend to forget about them, such as the quantization of color to 24 bits or the use of three-color channels red, green, and blue to represent the spectral response of the virtual scene. Even storing a polygonal mesh may involve simplification, since we clamp the precision of the coordinates and normals typically to 32 bits each for X, Y, and Z. Although useful and long-used approximations, especially in interactive graphics, the quantization of color and geometric attributes is indeed simplification, and recognizing this can lead to many benefits. In general the line between simplification and lossy compression techniques, which include quantization in its various forms, can be blurry. In the main, however, we will define LOD and simplification throughout this book as processes that reduce the complexity of polygonal meshes, not their precision or storage size. Shading and illumination form another spectrum of simplification in computer graphics. Many packages let the user control rendering performance with the familiar progression from wireframe to flat shading, to Gouraud shading, to texture mapping, and finally to per-pixel effects, such as Phong shading, bump mapping, and Team LRN 1. Lighting calculations range from simple local illumination techniques, such as the Phong model still used for the majority of interactive graphics, since this is what most hardware supports , to sophisticated global illumination methods such as radiosity and path tracing. Designers have long recognized the possibility of managing rendering performance by varying the complexity of shading and illumination applied to objects within the scene. The connection between texture mapping and level of detail deserves particular mention. For example, Dumont et al. A more frequent technique, however, replaces geometric detail in LODs with texture maps. Termed imposters by Maciel and Shirley [Maciel 95], these textured LODs are especially common in video games and visual simulation applications. Although we chiefly focus on geometric level of detail management, we discuss the use of textured imposters and backdrops in Chapter 6. We begin with an overview of three basic frameworks for managing level of detail: The original scheme proposed by Clark in and used without modification in most 3D graphics applications today, this approach creates multiple versions of every object, each at a different level of detail, during an offline preprocess. At run-time the appropriate level of detail, or LOD, is chosen to represent the object. Since distant objects use coarser LODs, the total number of polygons is reduced and rendering speed increased. Because LODs are computed offline during preprocessing, the simplification process cannot predict from what direction the object will be viewed. The simplification therefore typically reduces detail uniformly across the object, and for this reason we sometimes refer to discrete LOD as isotropic or view-independent LOD. Discrete LOD has many advantages. Decoupling simplification and rendering makes this the simplest model to program: Furthermore, modern graphics hardware lends itself to the multiple model versions created by static level of detail, because individual LODs can be compiled during preprocessing to an optimal rendering format. For example, depending on the particular hardware targeted, developers Team LRN

10 Chapter 1 Introduction may convert models to use features such as triangle strips, display lists, and vertex arrays. These will usually render much faster than simply rendering the LODs as an unordered list of polygons. Rather than creating individual LODs during the preprocessing stage, the simplification system creates a data structure encoding a continuous spectrum of detail. The desired level of detail is then extracted from this structure at run-time. A major advantage of this approach is better granularity: This frees up more polygons for rendering other objects, which in turn use only as many polygons as needed for the desired level of detail, freeing up more polygons for other objects, and so on. Better granularity thus leads to better use of resources and higher overall fidelity for a given polygon count. Continuous LOD also supports streaming of polygonal models, in which a simple base model is followed by a stream of refinements to be integrated dynamically. When large models must be loaded from disk or over a network, continuous LOD thus provides progressive rendering and interruptible loadingâ€”often very useful properties. Thus view-dependent LOD is anisotropic: For instance, nearby portions of the object may be shown at higher resolution than distant portions, or silhouette regions of the object shown at higher resolution than interior regions Figures 1. This leads to still better granularity: This in turn leads to still better fidelity for a given polygon count, optimizing the distribution of this scarce resource. Indeed, very complex models representing physically large objects, such as terrains, often cannot be adequately simplified without view-dependent techniques. Creating discrete LODs does not help: The field of view is shown by the two lines. The model is displayed at full resolution near the viewpoint and at drastic simplification far away [Luebke 01a]. Unsegmented or poorly segmented data, such as the skeleton extracted from an MRI scan shown in Figure 1. Here almost the entire data set forms one connected surface, preventing the use of discrete LODs to speed rendering and preserving detail in the area of interest. Another source of difficult models is scientific visualization, which tends to produce extremely large data sets that are rarely organized into conveniently sized objects. Again, view-dependent LOD can enable interactive rendering without manual intervention or extra processing for segmentation. Team LRN a b Figure 1. Note that the polygons facing away from the viewer, shown here for clarity, would typically be culled away by backface culling. View-dependent LOD, despite its advantages, also comes with some significant drawbacks. The extra processing required for evaluating, simplifying, and refining the model at run-time, and the extra memory required by the view-dependent data structures, steer many developers away from this approach. Video games, often the most demanding and tightly optimized of graphics applications, have been notably reluctant to adopt view-dependent LOD outside of situations that almost require it, such as terrain rendering.

Chapter 5 : Level of Detail for 3D Graphics : Jonathan D. Cohen :

Level of Detail for 3D Graphics is absolutely a must-have book for practitioners in any graphics field including game programming, scientific or medical visualization, computer aided design, or virtual reality systems.

Please help improve it by rewriting it in an encyclopedic style. June Learn how and when to remove this template message The origin [1] of all the LOD algorithms for 3D computer graphics can be traced back to an article by James H. At the time, computers were monolithic and rare, and graphics were being driven by researchers. The hardware itself was completely different, both architecturally and performance-wise. The original algorithm presented a much more generic approach to what will be discussed here. After introducing some available algorithms for geometry management, it is stated that most fruitful gains came from " The same environment structuring is now proposed as a way to control varying detail thus avoiding unnecessary computations, yet delivering adequate visual quality: For example, a dodecahedron looks like a sphere from a sufficiently large distance and thus can be used to model it so long as it is viewed from that or a greater distance. However, if it must ever be viewed more closely, it will look like a dodecahedron. One solution to this is simply to define it with the most detail that will ever be necessary. However, then it might have far more detail than is needed to represent it at large distances, and in a complex environment with many such objects, there would be too many polygons or other geometric primitives for the visible surface algorithms to efficiently handle. The proposed algorithm envisions a tree data structure which encodes in its arcs both transformations and transitions to more detailed objects. In this way, each node encodes an object and according to a fast heuristic , the tree is descended to the leaves which provide each object with more detail. The significant point, however, is that in a complex environment, the amount of information presented about the various objects in the environment varies according to the fraction of the field of view occupied by those objects. The paper then introduces clipping not to be confused with culling although often similar , various considerations on the graphical working set and its impact on performance, interactions between the proposed algorithm and others to improve rendering speed. Well known approaches[edit] Although the algorithm introduced above covers a whole range of level of detail management techniques, real world applications usually employ different methods tailored to the information being rendered. Because of the appearance of the considered objects, two main algorithm families are used. The second considers the polygon mesh being rendered as a function which must be evaluated requiring to avoid excessive errors which are a function of some heuristic usually distance themselves. The given "mesh" function is then continuously evaluated and an optimized version is produced according to a tradeoff between visual quality and performance. Darker areas are meant to be rendered with higher detail. An additional culling operation is run, discarding all the information outside the frustum colored areas. Obtaining those models requires an external algorithm which is often non-trivial and subject of many polygon reduction techniques. Successive LOD-ing algorithms will simply assume those models are available. DLOD algorithms are often used in performance-intensive applications with small data sets which can easily fit in memory. Although out-of-core algorithms could be used, the information granularity is not well suited to this kind of application. This kind of algorithm is usually easier to get working, providing both faster performance and lower CPU usage because of the few operations involved. DLOD methods are often used for "stand-alone" moving objects, possibly including complex animation methods. A different approach is used for geomipmapping , [4] a popular terrain rendering algorithm because this applies to terrain meshes which are both graphically and topologically different from "object" meshes. Instead of computing an error and simplify the mesh according to this, geomipmapping takes a fixed reduction method, evaluates the error introduced and computes a distance at which the error is acceptable. Although straightforward, the algorithm provides decent performance. A discrete LOD example[edit] This section possibly contains original research. Please improve it by verifying the claims made and adding inline citations. Statements consisting only of original research should be removed. September Learn how and when to remove this template message As a simple example, consider a sphere. A discrete LOD approach would cache a certain number of models to be used at different distances. Because the model can

trivially be procedurally generated by its mathematical formulation, using a different amount of sample points distributed on the surface is sufficient to generate the various models required. This pass is not a LOD-ing algorithm. Visual impact comparisons and measurements Image.

Chapter 6 : Level of Detail

There are several kinds of LOD based on camera distance. Geometric, animation, texture, and shading variations are the most common (there are also LOD changes that can occur based on image size and, for gaming, hardware capabilities and/or frame rate considerations).

In a similar way to traditional maps, 3D models are an abstraction of the real world: The amount of detail that is captured in a 3D model, both in terms of geometry and attributes, is collectively referred to as the level of detail. In this article, the authors propose an improved specification for defining the level of detail in a 3D city model. For instance, a municipality will specify an LOD when tendering 3D modelling work to a company. This defines five LODs, ranging from a simple 2. Higher LODs do not only increase in their geometric complexity but also in their semantic richness, that is the description of the geometry. Practitioners actively use the LOD designations as shorthand of the specification and for expressing the fineness of a 3D model, and it has become a de facto standard even when models are not related to CityGML. However, the standard defines the LODs only narratively, without clear specification of the requirements for each. As a result, ambiguities and misunderstandings are possible. Hence, two models of significantly different complexities may still be considered as the same LOD. A prominent example is LOD2. In practice, if an LOD2 model is ordered, it is not certain if semantics are defined and if dormers are present, which might be important for the intended application Figure 2. Due to this shortcoming in the standard, the data behind an LOD2 model could be almost anything. This hinders the use and exchange of models in practice. In particular, it is difficult to estimate and compare costs if the definition of an LOD is not clear. LOD-defining Parameters The authors examined dozens of specifications of 3D data, internal practices of companies and tenders, and spoke with users about their views on the LOD concept. The LOD concept has been decomposed into six defining metrics, as follows: Thanks to this decomposition, and because each of the metrics can be quantified, it is possible to define the LOD unambiguously for each model. New Specification During the research, it became apparent that it is difficult to uniformly specify requirements for each of the six metrics. Different applications rely on different types of models, and thus the metrics may vary. The authors therefore developed a specification format Figure 3 intended for industry use for precise specification of the requirements prior to the acquisition of a 3D city model. Based on the framework developed, the authors constructed their own series of 10 precisely defined LODs which do not leave much ambiguity or gaps between them in order to address the shortcomings of the CityGML concept. Standardisation The developed framework enables each stakeholder to define their own series of LODs. It is hoped that this will lead to unambiguous specifications and clear procurement of 3D models, eliminating many potential misunderstandings. OGC has also recognised the need for refining the concept, and relevant efforts are already underway. The authors of this article are involved in the CityGML Standards Working Group which is currently developing the new version of the standard, due in Previously he worked in business development at Geofoto Zagreb, Croatia. He is particularly interested in combining the fields of GIS and computational geometry. She obtained her PhD degree 3D cadastre in Jantien combines her professorship with jobs as a researcher at both the Kadaster and Geonovum. Further reading Biljecki, F. Revisiting the concept of level of detail in 3D city modelling. Formalisation of the level of detail in 3D city modelling. Computers, Environment and Urban Systems, 48, pp. Spatio-semantic coherence in the integration of 3D city models. Modelling qualities in space and time. New concepts for structuring 3D city models â€” an extended level of detail concept for CityGML buildings. Ho Chi Minh City, Vietnam, pp.

DOWNLOAD PDF LEVEL OF DETAIL FOR 3D GRAPHICS

Chapter 7 : Level of Detail for 3D Graphics

Professional 3D services since including: custom 3d modeling, model conversion, texture map creation, and graphics design. Offers a library of 3D objects for sale, and free sample model, available in most formats.

Get Your Copy Here Excellent Tips For A Improve Ebook Reading Many of the times, it has been believed that the readers, who are using the eBooks for first time, happen to really have a difficult time before becoming used to them. There present variety of reasons behind it due to which the readers stop reading the eBooks at their first most effort to use them. Yet, there exist some techniques that could help the readers to have a nice and effective reading encounter. Someone should correct the correct brightness of display before reading the eBook. Because of this they suffer with eye sores and head aches. The very best option to overcome this severe issue is to reduce the brightness of the screens of eBook by making particular changes in the settings. A great eBook reader should be installed. It will be helpful to really have a good eBook reader to be able to truly have a great reading experience and high quality eBook display. You can also use complimentary software that can provide the readers that have many functions to the reader than just a simple platform to read the desirable eBooks. Besides offering a place to save all your precious eBooks, the eBook reader software even offer you a high number of characteristics as a way to improve your eBook reading experience compared to the conventional paper books. You can even enhance your eBook reading experience with help of alternatives provided by the software program like the font size, full display mode, the specific number of pages that need to be displayed at once and also change the color of the backdrop. You ought not make use of the eBook constantly for a lot of hours without rests. You must take proper rests after specific intervals while reading. Nevertheless, this does not mean that you need to step away from the computer screen every now and then. Constant reading your eBook on the computer screen for a long time without taking any break can cause you headache, cause your neck pain and suffer with eye sores and in addition cause night blindness. So, it is critical to give your eyes rest for a while by taking rests after specific time intervals. This will help you to prevent the troubles that otherwise you may face while reading an eBook constantly. While reading the eBooks, you need to favor to read huge text. Typically, you will note the text of the eBook will be in moderate size. It is proposed to read the eBook with enormous text. So, boost the size of the text of the eBook while reading it at the display. Despite the fact that this will mean you will have less text on each page and greater amount of page turning, you will have the ability to read your desirable eBook with great convenience and have a good reading experience with better eBook display. It is suggested that never use eBook reader in full screen mode. It is suggested not to go for reading the eBook in fullscreen mode. While it may look easy to read with full screen without turning the page of the eBook quite often, it put lot of strain in your eyes while reading in this mode. Consistently prefer to read the eBook in the same span that would be similar to the printed book. This is so, because your eyes are used to the span of the printed book and it would be comfy that you read in exactly the same way. Test out various shapes or sizes until you find one with which you will be comfortable to read eBook. By using different techniques of page turn you can also enhance your eBook encounter. You can try many strategies to turn the pages of eBook to enhance your reading experience. Check out whether you can turn the page with some arrow keys or click a particular portion of the screen, apart from utilizing the mouse to handle everything. Try using the mouse if you are comfortable sitting back. Lesser the movement you must make while reading the eBook better is going to be your reading experience. Specialized issues One difficulty on eBook readers with LCD screens is the fact that it will not take long before you strain your eyes from reading. This will definitely definitely help make reading easier. By using all these powerful techniques, you can definitely enhance your eBook reading experience to an excellent extent. This advice will help you not only to prevent certain risks which you may face while reading eBook regularly but also ease you to enjoy the reading experience with great relaxation. The download link provided above is randomly linked to our ebook promotions or third-party advertisements and not to download the ebook that we reviewed. We recommend to buy the ebook to support the author. Thank you for reading.

DOWNLOAD PDF LEVEL OF DETAIL FOR 3D GRAPHICS

Chapter 8 : Level of detail - Wikipedia

Level of detail (LOD) techniques are increasingly used by professional real-time developers to strike the balance between breathtaking virtual worlds and smooth, flowing animation. "Level of Detail for 3D Graphics" brings together, for the first time, the mechanisms, principles, practices, and.

Chapter 9 : Unity - Manual: Level of Detail (LOD)

In computer graphics, accounting for Level of detail involves decreasing the complexity of a 3D model representation as it moves away from the viewer or according to other metrics such as object importance, viewpoint-relative speed or position.