Multiresolution Foliage Rendering

C. Gasch[0000−0003−2013−2839] and M. Chover[0000−0002−0525−7038] and I. Remolar[0000−0002−7743−2579] and C. Rebollo[0000−0002−1328−2110]

Institute of New Imaging Technologies,
Universitat Jaume I, 12006 Castellón
{cgasch,chover,remolar,rebollo}@uji.es

Abstract. This work presents a continuous level of detail representation of foliage of trees. Multiresolution modeling allows to adapt the number of polygons to render to the relevance of the object in the scene. However, foliage is represented by isolated polygons, so most of the multiresolution modeling methods do not work properly with this part of the tree. This paper presents a multiresolution model that allows to adapt the number of leaves to the relevance of the foliage in the scene. The criterion to select the appropriate leaves to render is based on a previously performed view-driven simplification. To adapt this parameter in real time, data structures and the necessary algorithms that allow us to extract the appropriate number of polygons are presented. Some tests have been developed to evaluate the proposed solution and results show the good performance of the presented continuous level of detail.

Keywords: Multiresolution modeling · Foliage representation · Level of detail.

1 Introduction

Vegetation is an essential part of the outdoor environments in applications where realism is a must. However, the vast amount of polygons that are required to represent them makes their inclusion in interactive applications a problem. Level of detail modelling has been confirmed as a solution to adapt the number of polygons of the objects in real-time rendering of scenes. This technique manages object representation while maintaining an adequate frame rate in real-time rendering applications. Nevertheless, foliage of the plants are represented by isolated polygons, so most of the multiresolution models in the literature can not properly manage their representation. This problem does not occur when the solid part of the plants, the trunk and the branches, are considered because they are represented by continuous meshes.

According to Ribelles et al. [16], multiresolution models can be classified in discrete and continuous ones. The first group provides the application with a set of individual representations with different levels of detail. This technique has been widely used and it is included in some game engines, such as Unity [19] or Unreal Engine [6]. The continuous multiresolution models allow the application

to vary the number of polygons that represent the object in a smooth way [17]. They avoid popping effects because the change from one resolution to another one is made in a continuous way. Both groups of multiresolution models are based on simplification methods that have previously processed the geometry of the object. This simplification process is performed off-line and results are used to build the data structure that form the appropriate multiresolution models. Simplification methods differ in the metrics they are based on. Because of the fact that foliage is not represented by continuous meshes, the simplification methods have to be adapted to this kind of representation. Some works have been presented that deal with this part of the plants. They use metrics based on geometry [14, 4] or based on view-point driven metrics [7].

This paper proposes a continuous multiresolution model that allows representing the foliage and adapting the number of polygons according to their importance in the scene. The simplification method used to build this level of detail structure is based on the image-based simplification presented by Gasch et al. [7], that uses metrics based on information theory [18]. This method is based on the leaf removal operation and, to avoid the pruning appearance, some leaves that remain are resized. This last operation allows the visual look of the foliage to be maintained as their level of detail is reduced.

The proposed multiresolution model, as well as its level of detail extraction algorithm, has been developed to be easily included in current game engines in order to facilitate its use. A plug-in to be included in Unity 3D has been implemented, so the test and experiments to evaluate the method have been performed using this game engine. Obtained results show the good performance of the proposed level of detail model.

The paper is organized as follows. Section 2 considers the state of art related to the existing techniques in the multiresolution modelling adapted to the foliage. Section 3 briefly describes the model overview. Section 4 briefly reviews the simplification method this multiresolution model is based on and then it describes the data structure that stores the leaves of the whole representation. Next, Section 5 analyzes the rendering algorithms that perform the extraction of the appropriate number of leaves that form the foliage representation. Section 6 describes the test that have been performed and analyzes the results obtained, and finally, section 7 presents some conclusions and observations related to the future work.

2 Previous work

Some methods have appeared in the literature addressed to reduce the complexity of the sparse component of the trees and plants. They can be classified mainly in three groups, depending on the method they use to represent the foliage: images, points or polygons.

The first group uses image-based representation to perform real time rendering of plants [10, 12]. They considerably reduce the amount of required geometry, but they are usually used to represent distant plants due to their parallax effect.

Methods based onpoint/line representation usually combine polygons to construct hybrid models for trees. At close distances, the plant is represented by a polygon. With increasing viewing distance, branch meshes will transform into lines and leaves into points [3, 8].

The last group in the classification uses the polygons to represent the plants, regardless of the distance of the object to the viewer. However, different compression techniques are used to reduce the amount of data to visualize. They usually use different metrics to compute the off-line simplification that allows to build the level of detail structure. These simplification methods can be based on stochastic, geometry or image metrics. They condition the performance of the multiresolution model.

Traditionally, works in the literature perform a level of detail modelling by means of multiresolution models built from simplification methods based on geometric metrics [15, 13, 4]. However, due to the sparse distribution of the leaves in the foliage, the metrics have to be adapted to this kind of meshes to ensure good performance of the process. Remolar et al. [14] present a simplification method by collapsing points based on the Hausdorff distance, comparing two sets of points, using the shortest distance between a point and the set of points. Deng et al. [4] expand this method to simplify thin or broad leaves, integrating both computations in an only simplification framework.

Other works have appeared that apply stochastic simplification [2, 9] to invisible parts of the foliage for performing the real time rendering. Dong et al. [5] propose a hybrid representation (HR) of a simplified tree model, which allows to adaptively select simplified models according to the resolution of different devices. Zhang et al [20] present a method that deals with forest visualization. They represent trees by means of polygon meshes plus semi-transparent lines. Line models with different transparencies are instanced on the GPU by merging multiple tree samples into a single model.

Image based simplification has been widely analyzed in literature. However, most of the works deal with general meshes [11]. Gasch et al. [7] present in their last work a simplification method based on a viewpoint-driven metric that uses the mutual information in order to choose the leaf to prune. Moreover, this method avoids the pruned appearance of the tree because the error introduced every time a leaf is pruned is compensated: the size of the nearest leaf is altered to preserve the leafy appearance of the foliage. This simplification method has been the one used to develop the presented work. Results obtained from gasch et al. [7] allows the user to obtain different approximations of the foliage of the tree, obtaining a discrete multiresolution model. This work is addressed to obtain continuous level of detail from the leaf simplification sequence obtained by the application of the method presented in [7].

3 Model overview

The presented continuous multiresolution model allows to adapt the number of polygons that form the foliage in real time, according to its importance in the

scene. It is based on the Viewpoint-Driven Simplification (VDS) of Plant and Tree Foliage [7]. The model stores the data obtained from the VDS method in a data structure. As the viewer moves across the scene, vegetation elements update their detail based on the distance to the viewer and the size of the area they occupy on the final image. To do this, some algorithms have been developed that access to that data structure and adapt in real time the number of polygons rendered to the appropriate view conditions. Algorithms add or remove leaves from the current foliage representation and scale the rest that remain in the foliage, reducing the rendering time required to ensure the realism of the scene.

4 Data Structure of the continuous multiresolution model

The data structure of the continuous multiresolution model has been conditioned by the results obtained from the VDS simplifications model. So, a brief overview of this method is exposed next.

4.1 Review of the simplification algorithm

The VDS simplification method allows simplifying the sparse part of the trees and plants. To do this, it uses a metric based on a theoretical measurement of information called viewpoint mutual information (VMI) [18]. This metric measures the degree of correlation between a set of points of view and the object to be simplified. Using this metric, the method can get the error that occurs when it is performed a modification to the object by comparing the result given by the sum of all the points of view in the original object, with the modified one.

The quality of the results when obtaining this error depends on the number of points of view. To do this, the method uses a distribution of cameras placed at the vertices of a dodecahedron.

The simplification process is based on a pruning operation. To do this, in each iteration, the method searches for the leaf that produces the least error when it is eliminated. Once it is found, this leaf is removed and the process is repeated until the desired level of simplification is reached. After determining the leaf to be removed in each iteration, the closest leaf is searched for and scaled based on the error introduced when it is eliminated and on the size of the removed leaf, as shown in Figure 1. In this way the leafy appearance of the foliage is maintained.

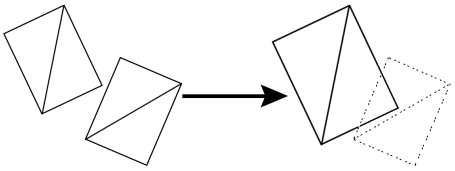

Fig. 1: Example of resizing a leaf that covers the pruned one in VDS

The results obtained by this method show that the use of the VMI metric generates a pruning leaf order that prioritizes those leaves that are hidden from any of the view-points used. Resulting from the simplification, foliage is emptied from the inside out and the possible gaps that may appear are filled by scaling the nearby leaves of the eliminated one.

4.2 Data structure

Analyzing the VDS simplification method, a data structure has been designed that stores all the data necessary to draw every one of the required level of detail. This structure is shown below.

```
Struct Leaf {
    Vertex vertices[4];
    int leafToSplit;
    int leafToCollapse;
    float scale;
}

vector<leaf> leaves;
```

The *vector leaves* stores the list of all the leaves that form the foliage. The order of storing the leaves is conditioned by the results obtained from the VDS algorithm, so it represents the drawing sequence.

All the data required to render a leaf is stored in the *struct Leaf*. It stores the data required both to render the original size and to render when some resizing is required. It is composed by:

- *vertices*: It stores the vertices that form the leaf.
- *leafToSplit*: The leaf that has to be modified by scaling its size when this leaf is added from the current representation.
- *leafToCollapse*: The leaf that has to be modified by scaling its size when this leaf is removed from the current representation.
- *scale*: The amount that must vary the size of the leaf to be modified.

5 Rendering algorithm

Once all the required data are stored in the multiresolution model, the algorithms that extract the appropriate level of detail have to be designed. They take into consideration the distance of the plant to the point of view as well as the size of that plant in the final rendered frame. The target of the criterion is to improve the quality of the render, so the closer the viewer is, the higher the realism has to be. As the camera moves far from the plant, fewer polygons are required to represent the foliage. The same case happens with the number of pixels used to render the final representation of the foliage: the importance of the plant in the frame conditions the number of leaves that is required to ensure the good quality of the final rendering.

These both criteria have been considered and codified in the formula shown in Equation 1, that obtained the value *LOD*. It combines the value *distanceToViewer*, obtained from the normalized distance (with a value between 0 and 1) between the center of the foliage and the camera position, and *projectedArea*, that stores the size of the projected area of the foliage in the current frame. Some constants (a, b) have been added to the equation to allow customizing the relevance of every one of the criteria in the equation. In the test, they have been set to 0.8 and 0.2 respectively, so the distance to the viewer has been considered more relevant.

$$LOD = distanceToViewer * a + projectedArea * b \tag{1}$$

This *LOD* value is normalized and allows to obtain the number of leaves to be finally rendered. Then, this number is compared with the one that is currently being rendered, performing different actions in the case the number of leaves to render is the same, fewer or higher than the currently number of leaves rendered. The actions to be performed are conditioned by this comparison, so:

- *Same number of leaves*: The level of detail to render is the same that is being currently rendered, so no actions are required to be performed.
- *Fewer leaves to render*: Some leaves have to be removed from the list to be rendered and some of the ones that remain have to be resized.
- *Higher number of leaves*: Some leaves have to be added to the final representation and some of the ones that are being rendered have to reduce their size in order to undo the scaling.

Algorithm 1 analyzes the process to follow in these cases. Every time the process starts, the algorithm has to evaluate the relevance of the foliage in the scene (*LOD*). This value is used to extract the number of leaves to render in this case *newLeaf*. Once this number has been obtained, it is determined if some leaves have to be added to the current approximation or some leaves have to be removed (*collapse*).

The scaling process is already calculated by the simplification method and saved in the structure used by this model to speed up the process and not have to recalculate the scaling on each leaf. Each leaf, when removed or added, only modifies another leaf (leafToModify), therefore, this model uses the leaves themselves to save the scale values that affect the other leaf, exchanging them when necessary. First, the model differentiates whether it is a *leafCollapse* or a *leafSplit*. In the first case, the scale value of the leaf to be modified is contained in the leaf to be collapsed. Therefore, both leaves exchange their scales as shown in Figure 2(a).

In the second case, as seen in the figure 2(b), the added leaf retrieves its original scale value from the nearby leaf. Which, in addition, also recovers its scale. When this process has been performed, the algorithm render the list of the leaves that form the current level of detail *(Draw(LOD))*.

Algorithm 1 Process to extract the Level of Detail

```
Function Extract(LOD) do
    newLeaf = totalLeaves * LOD;
    if newLeaf > currentLeaf then
      for i = 0; i < newLeaf − currentLeaf; i++ do
            leafSplit(currentLeaf);
            currentLeaf++;
      end for
    else
      for i = 0; i < currentLeaf − newLeaf; i++ do
            leafCollapse(currentLeaf);
            currentLeaf −−;
      end for
    end if
end function

Function leafSplit(currentLeaf)
    leafToSplit = leaves[currentLeaf].leafToSplit;
    aux = leaves[currentLeaf].scale;
    leaves[currentLeaf].scale = leaves[leafToModifiy].scale;
    leaves[leafToModifiy].scale = aux;
end function

Function leafCollapse(currentLeaf) do
    leafToCollapse = leaves[leaf].leafToCollapse;
    aux = leaves[currentLeaf].scale;
    leaves[currentLeaf].scale = leaves[leafToCollapse].scale;
    leaves[leafToCollapse].scale = aux;
end function
```

Algorithm 2 Process to render the Level of Detail

```
Function Draw(LOD) do
    lastLeave = totalLeaves * LOD;
    for i = 0; i < lastLeave; i++ do
      drawLeave(i);
    end for
end function
```

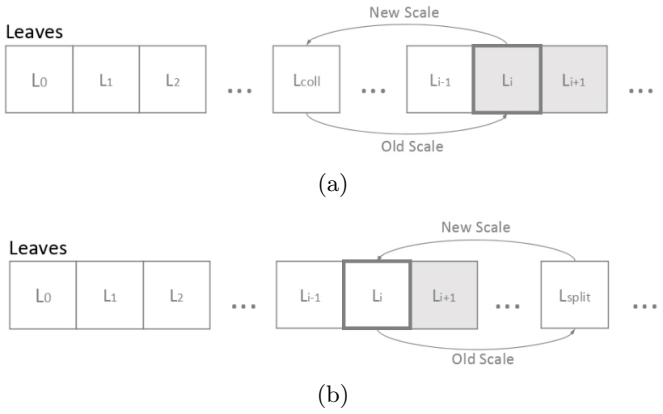

Fig. 2: (a) Collapse operation, removing Leaf L_i. (b) Split operation, adding Leaf L_i.

6 Results

The experiments have been performed using a computer with an Intel Core i7-3517U 2.40GHz processor with 8 GB RAM and AMD Radeon HD 8500M/8700M graphics card. All the geometric models of plants that have been used in the tests have been modeled using the commercial modeling tool Xfrog [35]. Unity 3D version 2018.2.7 was used to implement the model multiresolution and to view the result.

Three different vegetation species have been considered in the experiments: *English oak* with 20,146 leaves, *Carya illinoinensis* with 8,059 leaves and *Betula populifolia* with 12,140 leaves. The trunks and branches used have been simplified by the method presented by [1], but they do not change during the execution of the simplification process presented in this work.

In order to validate the multiresolution model, tests have been carried out with the three plant species separately and with forests formed by these species.

6.1 Tree Rendering

The performed test has analyzed each vegetable species individually. Every one of them has been represented and then the camera has been moved away from the model, adapting the level of detail of its representation according the formula previously presented. Then, the temporal cost of performing the visualization of the approximation have been calculated. This has been measured by obtaining the frames per second (FPS) while the camera moves around a circular path that surrounds the model. This circular path has increased the radius for performing the experiments. The results obtained can be seen in Figure 3. As it can be observed in the chart, initially there is a great difference between the three vegetable species due to the different number of leaves. However, as the camera

zooms out, and therefore the level of simplification increases, this difference decreases, further increasing the frames per second.

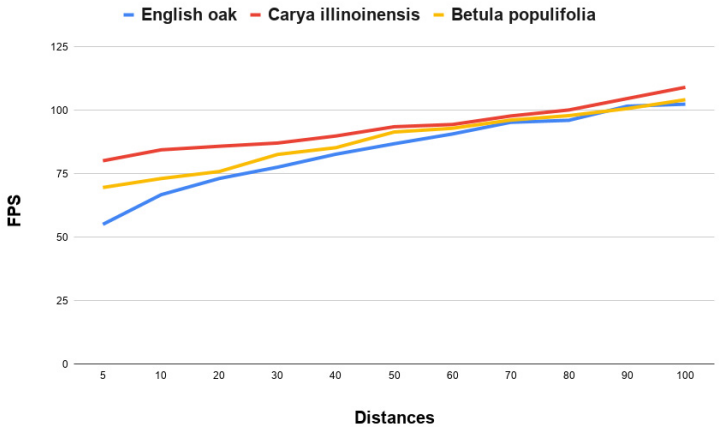

Fig. 3: Comparison of the FPS obtained with the three vegetable species at different distances.

*/

Regarding the time used to extract the appropriate level of detail, it has been measured in milliseconds per every distance. The obtained results can be seen in Figure 4. At the initial point, since the camera is close to the model and therefore there is no simplification, the cost is reduced to the render time itself. Then, as the method starts to reduce the detail of the foliage, the difference between the extraction of required approximation of the three vegetable species is greater. However, as the camera zooms out and the representations considerable reduce their number of leaves, the extraction cost decreases and the three species reduce the difference between them.

6.2 Forest Rendering

In order to better analyze the presented model, some tests have been carried out with groups of trees to represent forests. Figure 5 shows the results obtained when analyzing the spacial cost of visualizing them while the number of trees is increased. This cost has been measured by obtaining the frames per second (FPS) while the camera moves around a circular path that point the forest center. Each forest is made up of only one vegetable species that adapt their resolution according to the distance to the camera. However, in order to evaluate the good quality of the results, the forest represented with the vegetable species *English oak* has been tested twice: one test with all the geometry, without performing

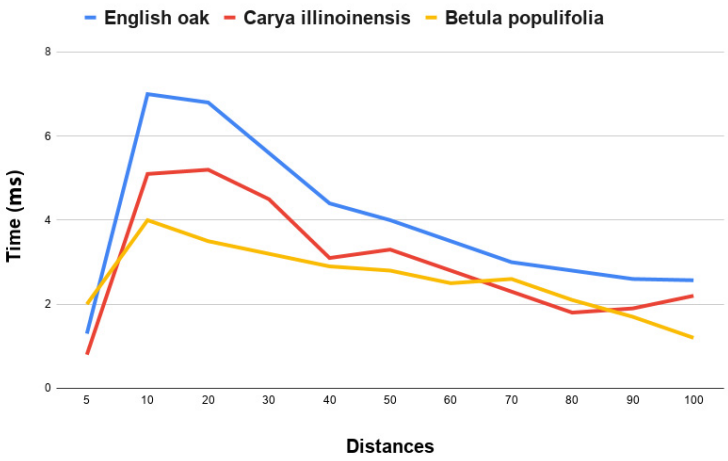

Fig. 4: Analysis of the time required to extract the different levels of detail for the three vegetable species at different distances.

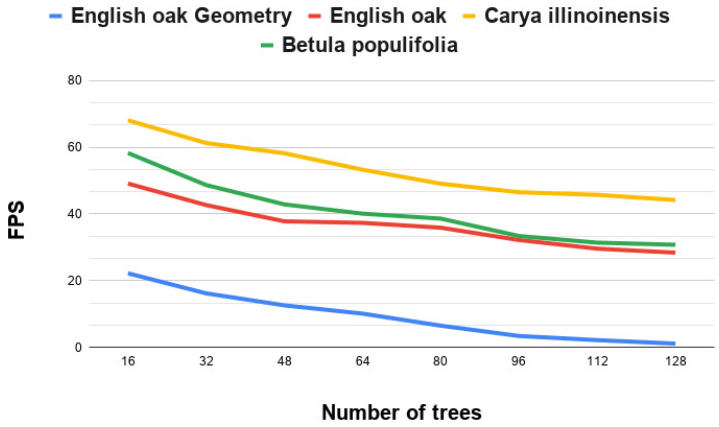

Fig. 5: Chart showing the FPS obtained while the amount of trees rendered increases.

reduction *English oak geometry*, and other one adapting the detail to the criteria *English oak*.

As it can be seen in the chart, the FPS considerably increases as the detail of the foliage of the rendered plants are adapted to the established criteria. Moreover, it can also be seen that the cost of extracting the appropriate leaves for every frame is not relevant in front of the wide reduction in the visualization time.

6.3 Visual results

In order to show how the presented multiresolution model allows simplifying the vegetation elements without reducing their visual quality, Figure 6 shows the simplification carried out at three different distances. For this, a tree with 20,000 leaves has been used, shown in 80% of its leaves, 30% and 15%.

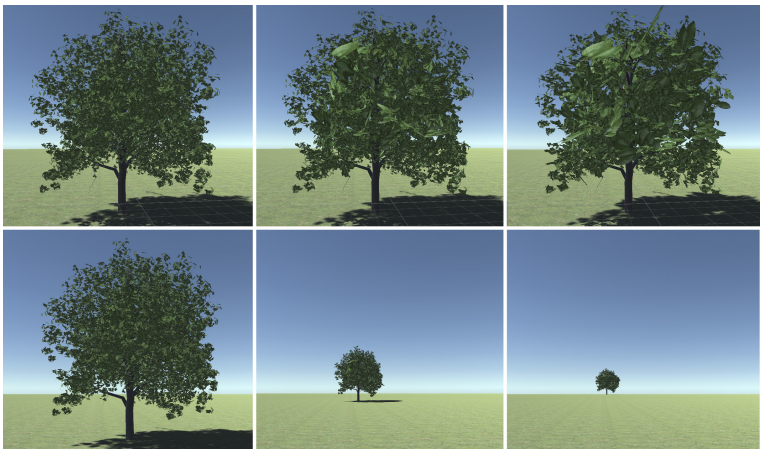

Fig. 6: Comparison between three different distances. With a reduction of 20%, 70% and 85% respectively.

In addition, a view of a small forest is shown in Figure 7. This forest contains nine trees with 20,000 leaves each one. After applying the simplification model, the number of total leaves drawn is 49,680 . However, the visual difference between the close and distant trees is barely appreciated.

7 Conclusions and future work

In this work, a multiresolution model for the foliage of plant elements has been presented, based on the Viewpoint-Driven Simplification method. This simplification method establishes a leaf removing order from the most hidden leaves

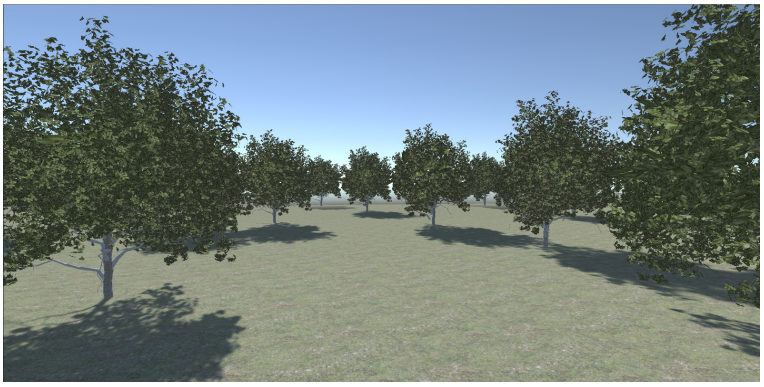

Fig. 7: Forest with nine trees with simplification applied

to the outside of the foliage. Also, the method avoids the pruning appearance of the less detailed representation by filling the possible gaps in the foliage by scaling the leaves close to the pruned ones. In this way, the foliage can be highly simplified without being barely noticeable by the viewer when the camera is far from the plant. Data structure and the algorithms that allow to extract the appropriate approximation have been presented in the paper.

Also, the multiresolution model have been designed to be easily implemented in game engines. The extraction time of the appropriate level of detail is very low being, as results demonstrate, less than 1 millisecond in models with more than 40,000 polygons. The results obtained demonstrate that the model allows to adapt in a continuous way the detail of the foliage in a scene. Regarding forest visualization, also results support that the rendering cost is significantly reduced while maintain the visual quality.

As future work, some animation is going to be applied to the leaves in order to simulate the effects of wind on them. In this future line of work, also trunk and branches are going to be animated adding some skeleton to the vegetable species.

Acknowledgments This work was supported by the Spanish Ministry of Science and Technology (Project PID2019-106426RB-C32) and the Universitat Jaume I research projects (UJI-B2018-56).

References

1. Castelló, P., Sbert, M., Chover, M., Feixas, M.: Viewpoint-driven simplification using mutual information. Computers Graphics **32**(4), 451 – 463 (2008). https://doi.org/https://doi.org/10.1016/j.cag.2008.05.005, http://www.sciencedirect.com/science/article/pii/S0097849308000551

2. Cook, R.L., Halstead, J., Planck, M., Ryu, D.: Stochastic simplification of aggregate detail. ACM Trans. Graph. **26**(3), 79 (2007)
3. Dachsbacher, C., Vogelgsang, C., Stamminger, M.: Sequential point trees. In: ACM Transactions on Graphics. pp. 657–662 (2003)
4. Deng, Q., Zhang, X., Yang, G., Jaeger, M.: Multiresolution foliage for forest rendering. Comput. Animat. Virtual Worlds **21**(1), 1–23 (2010)
5. Dong, T., Liu, S., Xia, J., Fan, J., Zhang, L.: A time-critical adaptive approach for visualizing natural scenes on different devices. PLoS ONE **2**, 1–26 (2015)
6. Epic Games: Unreal engine, Available online:http://www.unrealengine.com/
7. Gasch, C., Remolar, I., Chover, M., Rebollo, C.: Viewpoint-Driven Simplification of Plant and Tree Foliage. Entropy **20**(4), 213 (Mar 2018)
8. Gilet, G., Meyer, A., Neyret, F.: Point-based rendering of trees. In: E. Galin, P.P. (ed.) Eurographics Workshop on Natural Phenomena (2005)
9. Gumbau, J., Chover, M., Remolar, I., Rebollo, C.: View-dependent pruning for real-time rendering of trees. Computers Graphics **35**, 364–374 (04 2011). https://doi.org/10.1016/j.cag.2010.11.014
10. Jakulin, A.: Interactive vegetation rendering with slicing and blending. In: de Sousa, A., Torres, J. (eds.) Proc. Eurographics 2000 (Short Presentations). Eurographics (2000), citeseer.ist.psu.edu/jakulin00interactive.html
11. Lindstrom, P., Turk, G.: Image-driven simplification. ACM Transaction Graphics **19**(3), 204–241 (2000). https://doi.org/http://doi.acm.org/10.1145/353981.353995
12. Lluch, J., Camahort, E., Vivo, R.: An image based multiresolution model for interactive foliage rendering. Journal of WSCG'04 **12**(3), 507–514 (2004)
13. Rebollo, C., Remolar, I., Chover, M., Gumbau, J.: Hardware-oriented visualisation of trees. In: Lecture Notes in Computer Science vol. 4263/2006. pp. 374–383 (2006)
14. Remolar, I., Chover, M., Belmonte, O., Ribelles, J., Rebollo, C.: Geometric simplification of foliage. In: Navazo, I., Slusallek, P. (eds.) Proc. Eurographics 2002 (Short Presentations). Eurographics (2002)
15. Remolar, I., Chover, M., Ribelles, J., Belmonte, O.: View-dependent multiresolution model for foliage. Journal of WSCG'03 **11**(2), 370–378 (2003)
16. Ribelles, J., López, A., Belmonte, O., Remolar, I., Chover, M.: Multiresolution modeling of arbitrary polygonal surfaces: a characterization. Computers & Graphics **26**(3), 449–462 (2002)
17. Ribelles, J., Chover, M., López, A., Huerta, J., Castellón, I.: Frame-to-frame coherence of multiresolution ordered meshes (1999)
18. Sbert, M., Feixas, M., Rigau, J., Viola, I., Chover, M.: Applications of information theory to computer graphics. In: Eurographics (2007)
19. Unity: Unity 3d. Available online:http://www.unity.com/ (2020), last accessed July 2020
20. Zhang, X., Bao, G., Meng, W., Jaeger, M., Li, H., Deussen, O., Chen, B.: Tree branch level of detail models for forest navigation. Computer Graphics Forum **36**(8), 402–417 (2017)