variable wind field. By translating all grass blade vertices, Boulanger et al. [4] simulated the same distortions of grass under the influence of wind field. Li et al. [5] simplified grass into a line of rigid sticks linked together by spherical joints. Physically based method is used to make the above-motioned simulation skeleton deformable. Wang et al. [6] adopted physically based method to calculate the deformation of grass in wind, and achieved realistic visual effect. Nevertheless, as the computation is based on Hooke's law, when wind force changes significantly, the animation sequence may not be very smooth. To tackle this problem, Qiu et al. [7] developed a combined scheme to handle large-scale grassland motion under the influence of wind field. The other algorithms and more specific surveys can be found in [8].

For the simulation of grass-object interaction, only a few studies have been addressed. Guerraz et al. [9] presented a method to allow virtual objects or characters to crush the grass. Nonetheless, it is a procedural approach, and there is no possibility to react to collision, based on the object's geometry. Niu [10] presented a method to perform grass-object interaction. Nevertheless, physically based approach leads to a high computation cost. Zhao et al. [11] proposed GPU-based method to simulate grass-object interaction. However, their collision detection algorithm is based on bounding box, high fidelity is unavailable. In addition, the grass-grass interaction is not considered. Orthmann et al. [12] put forward a GPU-based approach to model responsive grass. They perform the simulation using a spring model, but in areas where grass is planted sparsely, the visual impression could be improved. Chen et al. [13] proposed procedural method to handle the grass-object interaction. Nevertheless, the grass is only suited to low view, due to the adoption of billboard. Most recently, Belyaev et al. [14] presented an integrated solution for animation, collision and rendering of grassland. Nonetheless, it does not perform explicit collision detection among the grass, and therefore reduces the fidelity of the rendering results.

3 Overview of Our Method

It is observed that when moving object enters a grass field, it collides with some grasses and the grasses that collide with object will bend. Meanwhile, the bending is propagated to other neighboring grasses due to the grass-grass interactions. After the objects passed, a track will be left on the field. Thus, to simulate grass interacting with object realistically, there are two issues must be considered. Firstly, when grasses collide with object, they should bend in the direction of external force. Secondly, the bending grasses will influence the surrounding grasses; grass-grass interaction must be taken into account. To realize the above effects, we have proposed a new approach. In this section, we give an overview of our method.

3.1 Level of Detail Representation of Grass Field

To achieve realistic simulation of grass motion under the influence of dynamic object, the modeling of grass field must be considered. Based on our previous work [7], we define three levels of detail for grassland, as shown in Fig.1.

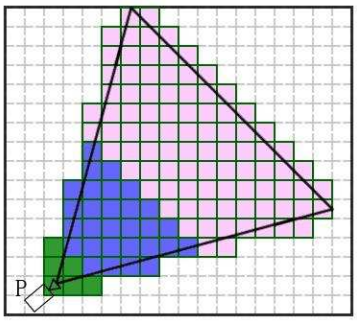


Fig. 1 Level of detail representation of grasses

In Fig. 1, point P indicates the viewpoint. For grass blades that are near the viewer, geometry-based representation is adopted to depict grass with rich details. At mid-distance, billboard-based method is applied, while for the faraway grasses, 2D texturing approach is used.

To manage the whole scene effectively, block-based scheme is applied.

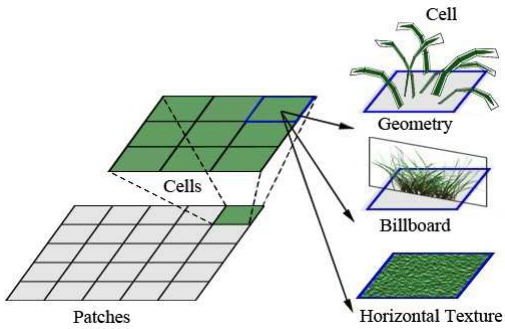


Fig. 2 Management based on blocks

As shown in Fig.2, we first divide the terrain into a set of square blocks with the same size, which are called *patches*. They are the unit of view frustum culling, i.e. the culling is implemented by intersection test based on patches, and we get rid of most of the patches that will not appear in the final image. Furthermore, each patch is subdivided into a number of uniform blocks, which are called *cells*.

They are the unit of representation, i.e. each cell determines the specific representation method of grass in itself.

The grasses those are close to the viewer have perceptible exterior characteristics. To show the features of grass with rich details, geometrically modeled grass blades are applied. A grass blade is split into several segments. Each segment is assumed to be rigid and can rotate around the joint that links itself to the adjacent segments.

Grasses at intermediate distance from the viewpoint always forms as clusters. Therefore, we treat a cluster of grass as one simulation element and use billboard to represent grass in this region. In each cell, there is a vertical quadrilateral billboard covered with semi-transparency grass pack texture. The length of each billboard is set to be longer than the boundary of the cell. Thus, numerous intersections occur between cells, preventing the creation of artifacts and offering more parallax effect.

For grasses those are far away from the viewpoint, detailed features and dynamic effects are imperceptible. Therefore, we apply classical 2D texturing approach that maps grass-like texture onto the ground to represent grass in these regions.

More details about the modeling of grassland can be found in [7].

3.2 Basic Principles of Our Approach

Based on the modeling methods of grass field, we have defined three kinds of deformable grasses: Direct-Bending-Grasses (DBG), Propagation-Bending-Grasses (PBG) and Billboard-Bending-Grasses (BBG). We deal with these diverse kinds of grasses separately.

DBG and PBG belong to geometry-based modeling grasses. When viewpoint approaches to grass field, the simulation of grass-object and grass-grass interactions are achieved by computing the bending of these two kinds of grasses.

DBG refer to the grasses that collide with dynamic object directly. We adopt GPU-based collision detection algorithm to determine the exact collision point. Physically based simulation approach is applied to calculate the bending of DBG. The bending of DBG is propagated to the neighbouring grasses (PBG) by grass-grass interaction. Obviously, PBG do not collide with dynamic object directly; their bending is affected by motion of surrounding DBG and neighbouring PBG.

BBG refer to billboard-based grass clusters, and the detail of each blade is imperceptible. Therefore, procedural method is adopted. The deformation of BBG is achieved by calculating the slope vector based on the distance from viewpoint to BBG.

For the grasses those are far away from the viewpoint, the detailed features and dynamic effects are all imperceptible. Consequently, animation is unnecessary.

The whole simulation process is divided into two stages that are off-line processing and real-time processing, as shown in Fig. 3.

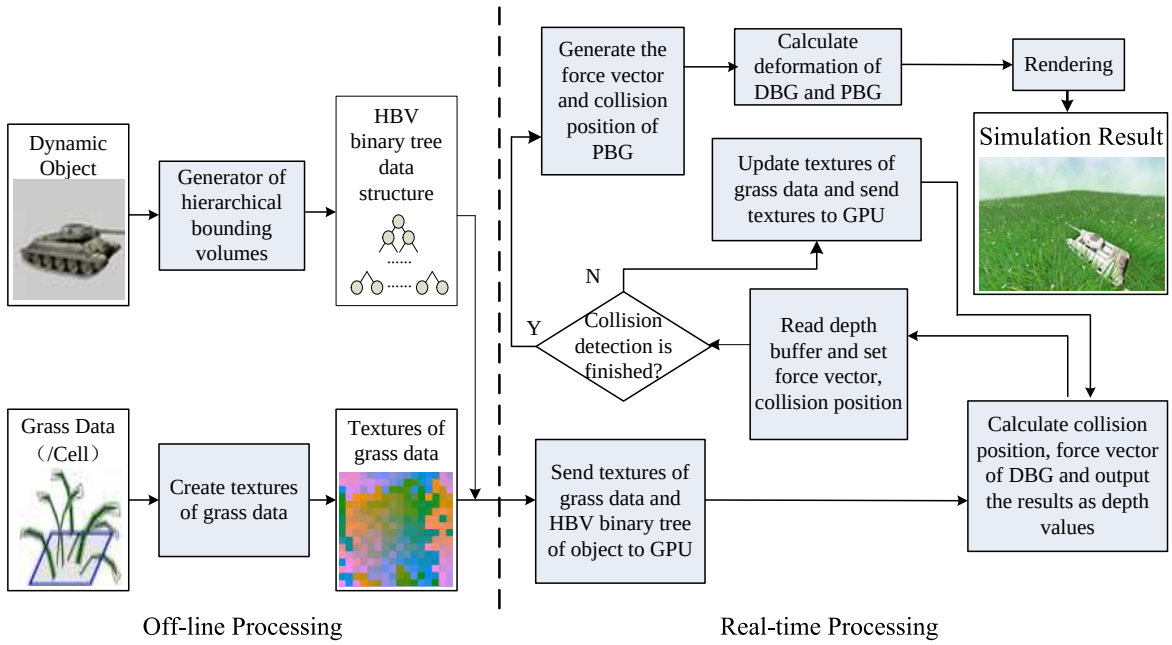


Fig. 3 Simulation framework

The main goal of off-line processing is to pre-process the dynamic object data and grass data. Collision detection, calculation of grass deformation and rendering are performed in real-time processing. The details of aforementioned two stages are detailed as follows:

(1) Off-line Processing: Firstly, we construct the hierarchical bounding volumes (HBV) of the dynamic object model. Then, based on the unit of patch, we deal with grass data (vertex coordinates of skeleton line) and organize them to textures. In other words, we store the vertex coordinates of skeleton line in textures.

(2) Real-time Processing: For DBG, the pre-processing data, which include textures of grass data and HBV of dynamic object, are sent to GPU. By using fragment program to implement collision detection on GPU, we can obtain the colliding position and force vector. For PBG, a propagation model of force is proposed to gain the approximate external force on PBG. In the processing of calculating the deformation, based on different kinds of deformable grasses, physically based and procedural approaches are applied respectively.

4 Real Time Collision Detection based on GPU

Inspired by the algorithm in [15] and combined with the features of grass blade, we implement precision controllable collision detection for DBG. The main process involves three stages:

(1) Construct HBV binary tree of dynamic object and organize grass data to textures based on patch.

(2) Perform intersection tests between dynamic object and patches. Only patches that intersect with object are taken into account. This can greatly reduce the number of grass blades involved in the collision detection.

(3) Traverse all the patches that intersect with object. For each patch, intersection tests between grass blades and the root node of HBV are carried out firstly. The results are output as depth values. Read depth buffer and get the collection of grass blades that intersect with object. The empty collection depicts no grass blades in current patch intersect with object. Otherwise, re-organize grass data to textures; a second and more exact collision test is performed. This procedure can be operated recursively, until achieving satisfactory precision or reaching the leaf nodes of HBV binary tree.

4.1 Organization of Grass Data

Each of the grass blades, which are close to the viewpoint, is split into several segments. Every segment is assumed rigid and can rotate around the joint that links itself to the adjacent segments. We use two-sided quadrilateral strips to approximate grass blades. Skeleton line data of grass blades are organized to textures for collision detection, i.e. the three-color channels of each texel in the texture are used to store a vertex coordinate of skeleton line.

Assume that a patch is composed of m grass blades, and each grass blade has n segments. $n+1$ textures are required to store grass data. As shown in Fig. 4, texture $k(k=0,1,2,...,n)$ stores k-th vertex coordinate of all the grass blades in the corresponding patch. The position of skeleton line data in texture (i, j) corresponds to the position of grass blade in patch (i, j). The width and length of texture can be expressed as follows:

$$w = h = \left|\sqrt{m}\right| \tag{1}$$

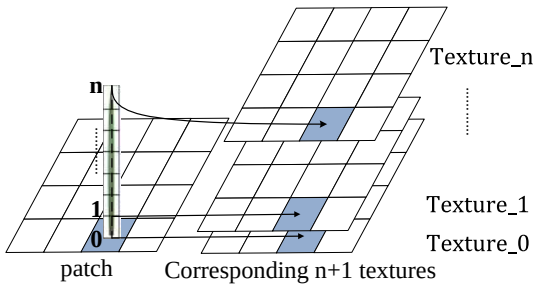


Fig. 4 Organization of grass data

4.2 Collision Detection between DBG and Object

GPU and CPU cooperate with each other to perform the collision detection. Fragment program implements intersection tests between grass blades and HBV binary tree of object. CPU obtains the results by reading the depth buffer, re-organizes the data of grasses that intersect with object to textures and sends them to GPU for iterative calculation of intersection tests. Besides, CPU takes charge of tasks related to precision control of collision detection.

Two kinds of data are required during the intersection tests between grass blades and HBV binary tree of object:

(1) The information of sides of skeleton line (assume each blade has been divided into n segments). Firstly, we create a viewport. The resolution of this viewport is the same as textures of the grass data. Then we construct a rectangle, which has the same size as viewport, and adopt parallel projection and texture mapping to map texture of

grass data to pixels of the rectangle. Therefore, we can get the mapping from each pixel of viewport to each grass blades in patch. The information of sides of skeleton line can be attained by texture sampling that achieved in fragment program.

(2) The data structure for defining HBV includes a centre point C, an orthonormal system based on a basis vector $[\vec{u},\vec{v},\vec{w}]$ and three values (*hu*, *hv*, *hw*) that describe half-width, half-height, half-length respectively.

At first, we take the root node of HVB binary tree as current node, using an intersection test algorithm between linear object and oriented bounding box [16] to implement intersection tests between grass blades and HBV in fragment program. The tests results are attained by reading depth buffer. If collisions have been detected, then we re-organize grass blades that intersect with object to smaller textures, and repeat the intersection test with the child nodes. More exact collision tests are performed subsequently. This procedure can be performed recursively, until achieving required precision or reaching the leaf nodes of binary tree.

As mentioned above, the precision of collision detection is related to iterations. When the value of depth is one, the judgment stops at the root node of HBV binary tree; it provides basic bounding box-based collision detection. Although the precision is proportional to the computational cost, we can control the precision based on actual application demand.

5 Force Calculation of Geometrical-based Grass

As mentioned in Section 3, DBG and PBG belong to geometrical-based grass. To simulate the force, we deal with these two kinds of grass respectively.

5.1 Force Simulation of DBG

The external force on DBG is affected by two factors. One is the equivalent normal vector $\vec{EqN}$, which belongs to the HBV binary node that has intersections with the grass. $\vec{EqN}$ reflects the force direction on DBG. Another factor is the velocity $\vec{v}$ of the dynamic object. The value of velocity is proportional to the magnitude of external force on DBG. Therefore, we depict the external force vector $\vec{EqF}$ on DBG as follows:

$$\vec{EqF} = \left|\vec{v}\right| \cdot \vec{EqN} \qquad (2)$$

$\vec{EqN}$ has a close relationship with the precision of collision detection. We calculate the weighted average of all the normal vectors of triangular surfaces that belong to the current HBV binary node. By using area of triangular surface as weight coefficient, we can get the corresponding equivalent normal vector.

Assume the number of triangular surfaces in a node is $i(i = 1,2...)$. The corresponding areas are $S_1, S_2, ..., S_t$. The normal vectors that correspond to triangular surfaces above-mentioned are $N_1, N_2, ..., N_t$. So, the equivalent normal vector of current node can be expressed as:

$$\vec{EqN} = \frac{S_1 \times \vec{N_1} + S_2 \times \vec{N_2} + ... + S_i \times \vec{N_i}}{S_1 + S_2 + ... + S_i} \qquad (3)$$

Multiple triangular surfaces within a node can be equivalent to one surface by using $\vec{EqN}$.

5.2 Generating Force Vector of PBG

The bending of DBG caused by grass-object interaction leads to the motion of surrounding PBG. However, modeling accurate grass-grass interactions requires a high computational cost. Instead of performing the collision detections among grasses, we proposed a propagation model to achieve the consecutive bending of grass and realize grass-grass interactions. Firstly, we give some definitions as follows:

Definition 1. *Energy*. It refers to the magnitude of the external force vector on a grass blade.

Definition 2. *Active status*. If the energy of a grass blade is large than the threshold value, the current grass blade is in active status.

Definition 3. *Termination status*. If the energy of a grass blade is smaller than the threshold value, it is in termination status.

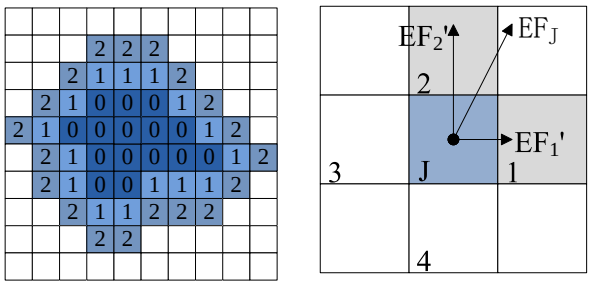
Definition 4. *Ancestor*. It refers to DBG.

Definition 5. *Children*. It refers to PBG. Ancestor propagates its force to the neighbouring PBG (the first generation children). The i-th generation children propagate their forces to the *i*+1th generation children.

Definition 6. *Adjacent relation*. A grass blade has adjacent relations with four grass blades immediately above, below, to the left, and to the right.

Definition 7. *Parallel relation*. For *i*-th generation children G, its neighbours are neither ancestor nor k-th generation children ($k \le i$), they have parallel relation.

Fig.5 (a) shows the procedure of force vector propagating. The ancestors are represented as number 0. After the first time propagation, first generation children are produced, and they are in active status. Propagation continues, until all the children of some generation are in termination status.



(a)Force vector propagation (b)Principle of vector propagation

Fig. 5 Force vector generation of PBG

As shown in Fig.5 (b). Assume that child J of generation G has force vector EF_J. Its energy is $|EF_J|$. The threshold value is L. The propagation process of J can be depicted as the following steps:

Step 1. Estimate status of J. If it is in dead status, propagation is finished. Otherwise, go to step 2.

Step 2. The force vector on J are orthogonally decomposed to two vectors EF_1' and EF_2'.

Step 3. For the grass blades, which are not only pointed by EF_1' and EF_2', but also have adjacent relations with J, they are judged whether they have parallel relations with J. If it is true, propagation is over. Otherwise, the force vectors of the two blades are expressed as follows:

$$\begin{cases} EF_1 = EF_{1origin} + EF_1' \cdot att \\ EF_2 = EF_{2origin} + EF_2' \cdot att \end{cases} \quad (4)$$

Parameter att ($att \in [0,1]$) is the decay factor, which is used to avoid endless propagation.

Step 4. Mark the two blades as children of generation $G+1$. Current propagation is over.

6 Computation of Deformation

6.1 Deformation of Geometrical-based Grass

We can get the collision information and external force on geometrical-based grass (both DBG and PBG) through the calculation of collision detection. According to these results, deformation of geometrical-based grass can be achieved by using the following steps.

Step 1. Calculate offset-base-vector

To implement consistent and stable performance during grass deformation, an offset-base-vector ($\vec{base}$) that reflects the whole deformation trend is required. All the other offset vectors, which control the deformation of grass vertexes, are based on $\vec{base}$.

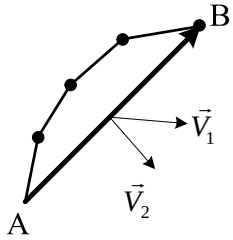


Fig. 6 Calculation of offset base vector

As shown in Fig.6, point A and point B represent the root and tip of a geometrical-based grass blade respectively. Vector AB, which is formed by connecting point A and B, reflects the bending trend of grass. The offset-base-vector that reflects deformation trend of grass blade, is perpendicular to $\vec{AB}$.In Fig.6, $\vec{V}_2$ is a unit vector that is perpendicular to $\vec{AB}$. $\vec{V}_1$ is the external force vector, which is attained by collision detection. Assume that *num* depicts the number of segments, which has intersections with the object. Therefore, offset-base-vector can be calculated as:

$$\vec{base} = \frac{(\vec{V}_2 \cdot \vec{V}_1)\vec{V}_2}{num} \quad (5)$$

From equation (5), we can see that the directions of vector $\vec{base}$ and vector $\vec{V}_2$ are the same. Furthermore, the magnitude of $\vec{base}$ is inversely proportional to the number of segment.

Step 2. Calculate offset vector of vertex

To achieve global stability of grass blades and avoid distortion during deformation, all the vertexes in the same segment use the same offset vector, and all the vertexes of a grass blade adopt an offset vector that has the same direction as $\vec{base}$. In addition, based on the observation of grass bending shape in nature, it is intuitive that the tip of grass blade has more offsets than the root. Thus, the offset vector corresponding to vertexes of segment i can be depicted as follows:

$$\vec{off}_i = P_i \cdot \vec{base} \quad (6)$$

where P_i is the weight, and its value is decided by weight function $P(x)$. The weight function can be expressed as:

$$P(x) = a\sqrt{1+\frac{x^2}{b^2}} - a \quad (7)$$

$x(x = 1,2,3...)$ is the corresponding segment number; a and b are constants.

Step 3. Length preservation

Using the offset vectors that mentioned in step 2 to update position of vertexes will lead to blade shape distortion. Length preservation must be taken into account. As shown in Fig. 7.

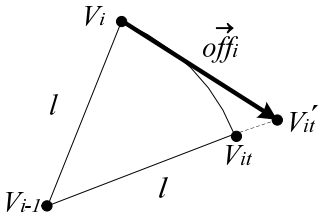


Fig. 7 Length preserving

Starting from the grass root point, the position of each vertex is updated. Influenced by offset vector $\vec{off}_i$, the new position of Vertex V_i in segment i is V_{it}'. Obviously, the length of $\overrightarrow{V_{i-1}V_{it}'}$ is different from $\overrightarrow{V_{i-1}V_i}$, which means the length of segment will change. Thus, depending on the direction of vector $\overrightarrow{V_{i-1}V_{it}'}$ and the unit length of a segment, V_{it} can be calculated.

6.2 Deformation of Billboard-based Grass

Billboard-based grass are at intermediate distance from the viewpoint, thus the deformation detail of each blade is imperceptible. Therefore, accurate collision detection is unnecessary.

In our strategy, whether to calculate the deformation depends on the results of intersection test between grass skeleton line and oriented bounding box (OBB) of the object. If skeleton line intersects with OBB, grass cluster deformation under the influence of dynamic object can be performed by controlling slope vector $\vec{slope}^b$. As shown in Fig. 8.

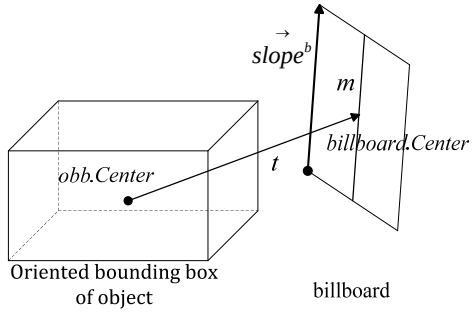


Fig. 8 Deformation of billboard-based grass

The slope vector can be expressed as:

$$\vec{slope}^b = \vec{slope}^b_{origin} + \vec{t} \cdot R \Big/ \left|\vec{t}\right|^2 \quad (8)$$

By connecting the centre of object OBB and the centre of billboard, vector $\vec{t}$ can be formed. The value of R is equal to the half-length of OBB diagonal.

From equation (8), we can see that the closer the distance between billboard and object, the greater slope the billboard has. It coincides with the real world.

6.3 Dynamic Transition

We adopt different deformation strategies for different representations of grass. To avoid popping artifacts, seamless dynamic transition scheme is indispensable. Two kinds of dynamic transition have been implemented.

(1) Transition from Polygon to Billboard

As shown in Fig. 9.

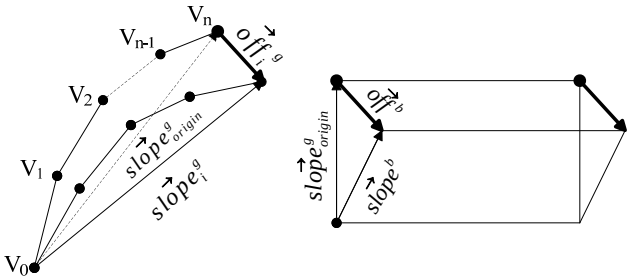


Fig. 9 Transition from polygon to billboard

According to the slope value of blade, transition from polygon to billboard can be achieved. The slope vector of grass i in current cell can be expressed as:

$$\vec{slope}_i^g = \vec{V}_n - \vec{V}_0 \quad (9)$$

The offset vector between current slope vector $\vec{slope}_i^g$ and original slope vector $\vec{slope}^g_{origin}$ can be formulated as:

$$\vec{off}_i^{\,g} = \vec{slope}_i^g - \vec{slope}^g_{origin} \quad (10)$$

We calculate the average offset vector in the patch, and set it to be the offset vector of billboard's slope vector. Assumed that the number of grasses in a patch is K, thus:

$$\vec{off}^{\,b} = (\sum_{i=1}^{K} \vec{off}_i^{\,g}) / K \quad (11)$$

$\vec{off}^{\,b}$ is used to update the slope vector of billboard.

$$\vec{slope}^{b} = (\vec{slope}^{b}_{origin} + \vec{off}^{b}) \cdot hight_g \quad (12)$$

where $\vec{slope}^{b}_{origin}$ is the original slope vector of billboard. $hight_g$ is the height of billboard. The deformation of billboard corresponding to geometrical-based grass can be achieved by using $\vec{slope}^{b}_{origin}$ to change the upper two vertexes of billboard.

(2) Transition from Billboard to Polygon

Assume the slope vector of billboard is $\vec{slope}^{b}_{t}$, it depicts the original position of billboard. Therefore, the offset of slope vector can be expressed as:

$$\vec{off}^{b} = \vec{slope}^{b}_{t} - \vec{slope}^{b}_{origin} \quad (13)$$

We use $\vec{off}^{b}$ as the external force vector $\vec{EqF}$ of geometrical-based grass and assume it only affects the last segment of grass blade. The transition from billboard to polygon can be implemented by using equation (5) to calculate the offset- base-vector.

7 Experiments and Discussions

We use OpenGL on C++ platform to implement the method presented in this paper. All the related experiments are carried out on a PC with Windows XP operating system, Intel Pentium 2.6GHz CPU, 2G memory, and NIVDIA 9800GT graphics card.

Firstly, we established a grass field, which can represent about 260,000 grasses. More specifically, the terrain with resolution of 512×512 is divided into 16×16 patches, each patch contains 32×32 cells, and each cell represents a grass blade. The value of collision detection precision is three. The rendering speed ranges from 23fps to 47fps. Fig. 10 and Fig. 11 illustrate the rendering results, which have little perceivable loss in image quality.

Fig. 10 Interaction between grasses and spherical object

Fig. 11 Interaction between grasses and object with complex shape

In order to testify the influence of collision detection precision on rendering efficiency, we carried out a series of pressure tests.

We did seven experiments with different value of collision detection precision from one to seven. The scale of grass field and the representation grass number in each cell are similar as previous test. The moving path of object is predefined. In each experiment, 5000 frames are sampled respectively. We achieved the lowest frame rate, highest frame rate and calculated the average frame rate. Fig. 12 shows the result.

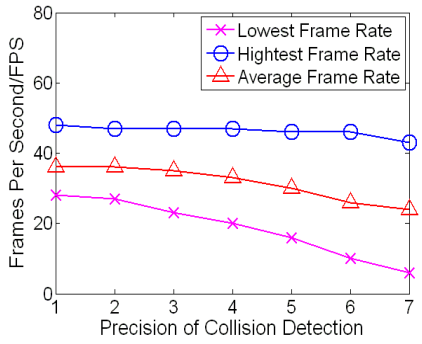
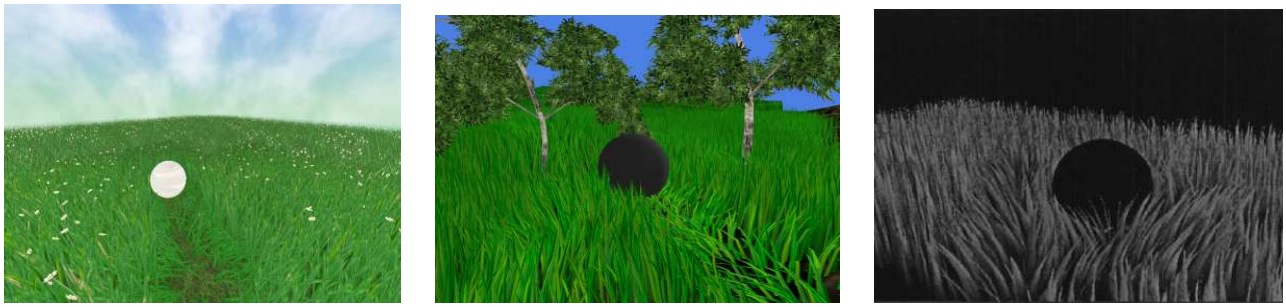


Fig. 12 Rendering efficiency with different precision

As shown in Fig.12, with the increase of collision detection precision, the rendering efficiency decreased. However, thanks to the multi-level representation of grass field and the combination with dynamic simulation method, the computational cost can be controlled effectively. When the precision of collision detection is 7, the average rendering speed is still 24fps. In general, when the precision of collision detection is between 3 and 5, it is sufficient to meet the requirement of visual quality. In fact, according to the application requirements, we can vary the precision.

Moreover, we compared the rendering performance of our approach with some existing methods [9-11, 13]. Fig. 13 and Fig. 14 illustrate the rendering results of various methods.



(a) Our method (b) Method in [9] (c) Method in [10]

Fig. 13 Performance comparison of different methods for the simulation of interaction between grass and spherical object

(a) Our method (b) Method in [11] (c) Method in [13]

Fig. 14 Performance comparison of different methods for the simulation of interaction between grasses and object with complex shape

As shown in Fig. 13 and Fig. 14, one typical advantage of our method is the vivid performance of details. After all, achieving a high performance is one of the major aims to real-time applications.

Furthermore, we compared features of our method with some previous works as mentioned in Section 2. The results are demonstrated in Table 1.

Table 1. Comparison of features with different methods

Method	Representation of grass	Precision of Collision Detection	Limitation of Dynamic Object	Grass-grass Interaction	Dynamic Transition
Guerraz et al. [9]	Three levels of detail	×	Unlimited	×	×
Niu [10]	Particle system	Low precision	Only spherical object	√	×
Zhao et al. [11]	Three levels of detail	Low precision	Unlimited	×	×
Orthmann et al.[12]	Billboard	Low precision	Unlimited	×	—
Chen et al. [13]	Billboard	Low precision	Unlimited	√	—
Our method	Three levels of detail	Variable precision	Unlimited	√	√

Although [9, 11] applied the same multi-levels representations of grass field as ours, our method has more advantages over them. Authors in [9] adopted a scalar field with a finite radius of influence around it to simulate grass-object interaction. It is a pure procedural method without collision detection, which leads to a limited quality of appearance. The researchers in [11] adopted OBB-based collision detection algorithm, which only provides a low precision of collision detection.

Moreover, grass-grass interaction and dynamic transition is not considered. In [10], physically based strategy is adopted, and grass-grass interaction is taken into account. However, the disadvantages of it include high computational cost and the limitation of dynamic object shape. In [12], grass field was modeled by geometry-based method and the grass-object collision detections are considered. However, the precision of collision detection can not be controlled, and the grass-grass interactions are ignored. In [13], grass-object and grass-wind interactions have been taken into account. Nevertheless, precision of collision detection is not variable. Besides, in a highly dense grass field, billboards may intersect with each other, due to lack of explicit collision detection.

8 Conclusion

In this paper, we have proposed a novel approach to simulate the interaction between grass and dynamic object, which achieves a better perception of interactive environments and is suitable for applications such as outdoor training simulations, driving simulator and games.

The major contributions of our work can be listed as follows:

(1) Physical and procedural simulation approaches are applied, according to the different levels of detail of grassland, which reduce the rendering cost and still allow high fidelity.

(2) To achieve consistency animation between different representations of grasses, a novel dynamic transition strategy is presented, which avoids popping artifacts appearing.

(3) According to the features of grass-object interactions in real world, an approach to simulate consecutive bending of grass caused by grass-object interaction is presented to enhance the reality of the simulation.

In the near future, we expect to implement the recovering process of grasses in response to the object. Furthermore, some other challenging issues are still seldom involved, such as lighting [17], burning, withering [18] and interacting with raindrops [19]. Performing and incorporating these effects into our framework is an interesting topic.

Acknowledgements:

The authors wish to thank the anonymous reviewers for their thorough review and highly appreciate the comments and suggestions, which significantly contributed to improving the quality of this paper. Also, we would like to thank Dr. Hao Fu who spent time on reading the earlier drafts of this paper.

This work is supported by National High Technology Research and Development Program of China (No. 2012AA011503), Project on the Integration of Industry, Education and Research of Guangdong Province (No. 2012B090600008) and the pre-research project (No. 51306050102).

References:

[1] F. Perbet, M. P. Cani, Animating prairies in real-time, *In Proc. I3D'01*, North Carolina, USA, 2001, pp. 103-110.

[2] K. Pelzer, GPU Gems: Rendering countless blades of waving grass, Addison-Wesley, 2004, pp. 107-121.

[3] B. Bakay, W. Heidrich, Real-time animated grass, *In Proc. Eurographics'02 (short paper)*, Saarbrucken, Germany, 2002.

[4] K. Boulanger, S. N. Pattanaik, K. Bouatouch, Rendering grass in real time with dynamic lighting, *IEEE Computer Graphics and Applications*, Vol.29, No.1, 2009, pp. 32-41.

[5] C. F. Li, X. Y. Guo, S. L. Lu, et al, Real-time simulation of meadow, *In Proc. SSSC'08*, Beijing, China, 2008, pp. 1205-1207.

[6] C. B. Wang, Z. Y. Wang, Q. Zhou, et al, Dynamic modeling and rendering of grass wagging in wind, *Computer Animation and Virtual Worlds*, Vol.16, 2005, pp. 377-389.

[7] H. Qiu, L. T. Chen, J. X. Chen, Dynamic simulation of grass field swaying in wind, *Journal of software*, Vol.7, No.2, 2012, pp. 431-439.

[8] H. Qiu, L. T. Chen, J. X. Chen, et al, Survey on realistic simulation of grassland, *Journal of Computer-Aided Design & Computer Graphics*, Vol.22, No.9, 2010, pp. 1628-1638.

[9] S. Guerraz, F. Perbet, D. Raulo, et al, A procedural approach to animate interactive natural sceneries, *In Proc. CASA'03*, New Jersey, USA, 2003, pp. 73-78.

[10] L. X. Niu, Research on the technologies of realistic graphics algorithm and real time rendering, *Master dissertation*, Capital Normal University, China, 2007.

[11] X. K. Zhao, F. X. Li, S. Y. Zhan, Real-time animating and rendering of large scale grass scenery on GPU, *In Proc. Itcs'09*, KIEV, Ukraine, 2009, pp. 601-604.

[12] J. Orthmann, C. Rezk-salama, A. Kolb, GPU-based responsive grass, *Journal of WSCG*, Vol.17, 2009, pp. 65-72.

[13] K. Chen, H. Johan, Real-time continuum grass, In Proc. VR'10, Walatham, USA, 2010, pp. 227-234.
[14] S. Belyaev, I. Laevsky, V. Chukanov, Real-time animation, collision and rendering of grassland, *In Proc. GraphiCon'2011*, Moscow, Russia, 2011, pp. 1-4.
[15] Z. W. Fan, H. G. Wan, S, Gao, Streaming collision detection using programmable graphics hardware, *In Proc. CAD & Graphics'03*, Macau, China, 2003, pp. 1-10.
[16] J. S. Philip, H. E. David, *Geometric tools for computer graphics*, Elsevier, 2003.
[17] K. Boulanger, S. N. Pattanaik, K. Bouatouch, Rendering grass in real time with dynamic lighting, *Computer Graphics and Applications*, Vol.29, 2009, pp. 32-41.
[18] S. H. Jiao, W. Wu, P. A. Heng, et al, Using Time-varying texels to simulate withering grassland, *Computer Graphics and Applications*, Vol.23, No.1, 2012, pp. 78-86.
[19] M. Yang, L. S. Jiang, X. S. Li, et al, Interactive coupling between a tree and raindrops, *Computer Animation and Virtual Worlds*, Vol.23, 2012, pp. 267-277.