Keywords: Page scheduling, Flight simulation, Branch-and-bound search.

Abstract

To address the issue of unpractical and inefficient terrain page searching in terrain scheduling of flight simulations, we studied the scheduling problem for further optimization. Firstly the page format organization using the holding-boundary idea was introduced, which well disposed of the rendering sub-problem in scheduling. Then based on the organized pages, we proposed the branch-and-bound search algorithm to quickly search necessary terrain pages in flight scenes without redundant data. The algorithm regarded searching pages as searching optimal results in a solution space tree or graph, and it could be applicable to both query and prefetching sub-problems in scheduling. Experiment results showed that, compared with previous methods, our search algorithm could significantly improve searching efficiency but consume less memory space. Hence our method could offer better terrain page scheduling and smoother large-scale terrain visualization in flight simulations.

1 Introduction

Virtual environment in flight simulations usually involves massive terrain data that is too large to be loaded into the main memory of a PC all at once. Therefore kinds of approaches were proposed to relieve this contradiction between the memory capacity and the visualization demand. And paging is a common approach at present. The idea of paging is to divide the whole terrain into small pages, and then load necessary ones for visualization. In this way loaded terrain data will not exceed the limit of the main memory, but still satisfy the visualization demand. Although paging provides a flexible way to realize large-scale terrain scene visualization, it invokes another problem, scheduling, to be discussed specially.

2 Related Work

The scheduling problem can be classified into three sub-problems: rendering, query and prefetching[1, 2]. Rendering is related to processing data to be displayed, and the processing could be on-line or off-line. On-line processing usually doesn't consider paging, and it's not fit for hardware nowadays because the workload of CPU and GPU is unbalanced[3, 4]. Off-line processing usually splits entire terrain data into pages and organizes them as a predefined format that is convenient for transport and displaying[5-11]. As much work has been done beforehand, terrain data can be rendered rapidly at runtime. Compared with on-line processing, off-line processing more corresponds to the paging idea since terrain data needs pre-processing of splitting and reorganization. The second sub-problem, query, is related to identifying and arranging data to be fetched or removed. That is more understandable in terms of page loading or unloading. Page loading should quickly search pages which are needed currently and prepare them for displaying. The shapes of fetching regions were presented to help search needed pages[1, 2], but detailed implementations were usually too simple or unclear. The priority queue was introduced to prepare pages quickly and properly[5, 6]. Page unloading should determine which pages are useless and remove them from the main memory. The least recently used (LRU) replacement strategy was applied during the determination[6-8]. Removing can also be implemented by an unloading priority queue. The prefetching sub-problem is related to fetching data that may be used in the near future. It consists of predicting future viewing state and fetching corresponding data. The former is the emphasis, and prediction methods are chosen according to specific situations[5,10,12]. The later is similar to the page loading in the query sub-problem, just fetching data based on predicted viewing states instead of actual states. Here lies the same issue of quickly searching necessary pages.

In this paper we will firstly introduce our format organization of terrain pages, which shows high rendering efficiency and seamless visual effect. Then based on the above organization, a page scheduling method that features the branch-and-bound search is proposed to address the problem of searching necessary pages in flight simulations.

3 Terrain Page Scheduling

3.1 Page Format Organization

There are different page format organization ways in off-line processing. One is to use the quadtree structure[5-10], which depends on CPU doing a lot of triangulation and crack avoiding work. The quadtree structure could provide good visual results but its manipulations are fussy. Another way is to use the hierarchy structure[13-15], which supports batched drawing of hardware and balances the
workload between CPU and GPU. But it can’t dispose of cracks between pages well. To make full use of GPU’s abilities and avoid unsatisfying cracks, we’ve introduced a kind of format organization based on the holding-boundary idea[16]. This idea can be explained by figure 1. For a regular square grid (RSG) in figure 1(a), previous simplification would remove boundary vertices as well as inner ones to get optimal simplifying, as in figure 1(b). However, this caused inconsistent boundaries between adjacent grids, and cracks emerged. If boundary vertices are held unchanged during simplifying, as in figure 1(c), cracks between grids shall be avoided without any extra work.

![Figure 1. Different simplification of RSGs.](image)

In [16] we have defined concepts of holding vertices and controlling vertices in a RSG. Besides we presented quantity range of holding vertices and simplification strategy of controlling vertices by improving Melax’s fast polygon reduction algorithm. Our simplification strategy could preserve terrain features better, and relieve the phenomenon of dense primitives caused by holding-boundary operations[17]. The core formulae of the simplification strategy are expressed as equation (1)(2):

$$\text{cost}(u, v) = \|u - v\|^2 \times C^2$$  \hspace{1cm} (1)

$$C = \max_{f \in F} \left\{ \sum_{g \in G_f} 0.5 \times \left(1 - n_f \cdot n_g \right) \right\}$$  \hspace{1cm} (2)

Equation (1) is to compute the cost of the vertex $u$ collapsing to the vertex $v$. Equation (2) is to compute the curvature value between $u$ and $v$, and more explanations can be found in [16].

![Figure 2. Hierarchy structure of a terrain page.](image)

Combining the RSG and its simplified results obtained via the holding-boundary idea, we organized the page format by the hierarchy structure, as figure 2 shows. The page format contains two main layers: the fine layer and the coarse layer. The original RSG constitutes the fine layer. The coarse layer further contains simplified results of varied extent, and the simpler the result is, the lower its level will be.

Our page format organization has three primary advantages:

1) Page data is saved properly. Firstly vertices information is recorded, and then vertex indices of different layers are saved. Using indices can avoid duplicated vertices information in multi-layers, thus reducing the page file size. Small size could save storage space and be fit for fast transport.

2) Because of the holding-boundary feature, no extra work shall be needed at runtime to ensure seamless splicing between terrain pages. The whole terrain scene shows good continuity in geometry, lighting and color space. Hence efficiency and visual effects are both assured.

3) This format organization could utilize GPU’s caching ability to accelerate drawing. Since vertices information is shared by multi-layers, it can reside in the vertex buffer. Only indices are transported and updated for terrain rendering, which decreases data exchanges.

This format organization based on the holding-boundary idea has provided a good solution to the rendering sub-problem. So much focus can be taken on other aspects in scheduling.

### 3.2 Branch-and-bound Search in Page Scheduling

As mentioned in related work, the query and prefetching sub-problem in scheduling have a similar issue: how to quickly search necessary pages for displaying. But practical implementations are usually too simple or unclear. In [1, 2], shapes of fetching regions are described as two forms: the fan shape and the circle shape shown in figure 3. But no content illustrates how to search pages within these two shapes’ scope.

![Figure 3. Shapes of fetching regions[1].](image)

![Figure 4. Kinds of frustums: having a short or narrow field of view.](image)
Studying in [7-9, 11] used frustums to help searching necessary pages, which has become a general method. But frustums they set had a short or narrow field of view, as shown in figure 4. Those kinds of frustums were not appropriate for the flight simulation, because high altitude will cause a distant and wide field of view. Thus pages covered by the frustum will not be as few as in figure 4, but involve large-scale data.

Analyses in [6, 10, 18] considered a larger field of view, but only [10, 18] gave a detailed method to search pages within the viewing scope. As shown in figure 5, the method in [10] estimated the maximum moving range of the frustum, and determined a fixed range of pages in horizontal and vertical directions. There are two defects in this method. Firstly in figure 5 the orientation of the frustum is ideal, but in practice it may rotate at any degree. Secondly, the start and the end of the fixed range of pages are difficult to determine as the frustum varies so much.

![Figure 5. Pages in the fixed range.](image)

The method in [18] projected the frustum onto the horizontal plane and found out necessary pages by judging the relationship between the projection area and pages. It was a good solution and much other work did in the same way. As the frustum may rotate, it’s better to extend the projection to a square area and then judge further, referring to figure 6.

![Figure 6. Extended area from the frustum projection.](image)

However, from figure 6 we can see that the extended area includes much redundant data that may burden the main memory. But removing those data using loop judgments is time-consuming. So we propose the branch-and-bound search algorithm to search needed pages directly, not via projection or other indirect way.

The branch-and-bound idea is to search the solution space of problems by the breadth-first or minimum-cost-first principle. It’s usually applied to problems that can be transformed to the issue of searching results in a solution space tree or graph. There are three key elements in a branch-and-bound algorithm, i.e. a search start point that makes the search begin, constraints that judge searched objects and branch-and-bound conditions that discard infeasible or non-optimum solutions in time.

Now thinking of searching pages, if we adopt the fan shape of the fetching region, as figure 7 shows, we can meet the three key elements in the branch-and-bound algorithm as following rules:

![Figure 7. Branch-and-bound search with the fan shape.](image)

1) The search start point : the view point O can help decide the start point for generally it’s upon terrain pages. The search just begins from the page under the point O.
2) Judgment constraints: the fan shape of the fetching region is the constraint that will judge whether a page is within the region. The judgments are expressed as equation (3)(4):

\[
\|P_O - P_{\text{page}}\| \leq D_{\text{threshold}} \tag{3}
\]

\[
\text{AngleBetween}(P_OV, P_OP_{\text{page}}) \leq \alpha_{\text{threshold}} \tag{4}
\]

where \(P_O\) and \(P_{\text{page}}\) are two-dimension horizontal positions of the view point and some page, and \(P_OV\) is the normalized view direction. \(D_{\text{threshold}}\) and \(\alpha_{\text{threshold}}\) are chosen thresholds for the distance and the angle.

3) branch-and-bound conditions: if a page meet the judgment constraints, four pages adjacent to it shall be tested by judgment constraints; whereas, four pages adjacent to it shall be regarded as not meeting constraints directly.

The pseudo-code of the branch-and-bound search is presented as following:

```csharp
Branch&BoundSearch( Page )
{
    if ( WithInFetchingRegion )
    {
        if ( NotBeLoaded )
            RequestForLoading;
        else
            LoadedState+1;
        TabThisPageAsSearched;
        if ( LeftPageIsNotSearch )
            Branch&BoundSearch( LeftPage );
        if ( RightPageIsNotSearch )
            Branch&BoundSearch( RightPage );
        if ( TopPageIsNotSearch )
            Branch&BoundSearch( TopPage );
        if ( BottomPageIsNotSearch )
            Branch&BoundSearch( BottomPage );
    }
}
```
else
TabThisPageAsSearched;

3.3 Supplementary Details for Page Scheduling
To facilitate the branch-and-bound search, two arrays are created with every array element using one byte memory space. One array records all pages’ loading states, i.e. loaded or unloaded. The other records pages’ search states, i.e. searched or not. These two arrays supply convenient references for judgments in the above pseudo-code. Moreover, the pages’ loading states array is used to implement the LRU replacement strategy. One byte memory space of the array element can counts from 0 to 255, so reasonable segmentation of this range is enough to indicate the pages’ using states.

We also do some speculative prefetching as [13] did. Past flight states could predict states in the near future to certain extent, because movements in flight simulations are stable and continuous at most time, seldom changing drastically. With predicted future states, prefetching can also be done efficiently using the branch-and-bound search.

Lastly, we bring multithreading in our program to make rendering smooth. The main thread is used to search necessary data and render scenes, and the second thread is special for loading needed data into or unloading useless data out of the main memory. In this way, lack of data will not lead to sudden pause and frame skipping.

4 Experiments
We performed experiments on the basis of our terrain format organization, so more attention can be paid to practical effects of search algorithms, not affected by visualization. The whole terrain area is 17.7 km by 14 km. The viewpoint moves in the same route at the speed of 60 m/s. The fan shape of the fetching region with the same parameters as in figure 7 is used in all experiments. If we extend the fan shape to a square shown in figure 6, the total pages that need to be loaded into the main memory shall be up to 750, taking too much space. So we use the loop judgment method removing redundant data. Then the branch-and-bound search is also tested and compared to the loop judgment.

Figure 8 shows numbers of pages in main memory changing with time when using the loop judgment and the branch-and-bound search. The number of the branch-and-bound search is a bit larger because the fetching region is larger than the frustum projection, but both numbers are smaller than that before removing redundant data.

Figure 9 shows the time spent for page searching every second. Although searching similar number of pages, the loop judgment spent about 10 times time than the branch-and-bound search. If prefetching is considered, that will be 20 times. So the branch-and-bound search algorithm is much more efficient. Figure 10 shows the top view of the terrain scene searched by our algorithm, which is similar to the fan shape of the fetching region.

Figure 10. Terrain pages searched by our algorithm.

5 Conclusion
To address the issue of unpractical and inefficient terrain page searching in terrain scheduling of flight simulations, we firstly introduced our terrain page format organization that was fit for scheduling and visualizing. So we could pay attention to query and prefetching sub-problem in scheduling. In these two sub-problems, how to quickly search necessary pages was not well implemented. Hence we proposed the branch-and-bound search algorithm to realize efficient and storage-saving page searching with the given shape of the fetching region. Meanwhile supplementary details were offered to dispose of query and prefetching sub-problems. Experiment results showed our algorithm could significantly improve searching efficiency but consume less memory space. So terrain page scheduling based on our branch-and-bound search is a more prac-
tical and efficient solution, making visualization smoother in flight simulations.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (Grant No. 61005067). And thank anonymous reviewers for helpful suggestions and comments.

References


