Term project for B4M39DPG (Data Structures for Computer Graphics)
17 - Hierarchical View-Frustum Culling for Z-buffer Rendering

Ales Koblizek (koblial2@fel.cvut.cz)

Department of Computer Graphics and Interaction,
Faculty of Electrical Engineering, CTU in Prague

13. 5. 2019

==

Contents:

==

1) project description

The goal of this project is to create an efficient implementation of a frustum culling algorithm for large static scenes, utilizing a bounding volume hierarchy, where the bounding volumes are axis-aligned bounding boxes.

First, a bounding volume hierarchy (BVH) is constructed using top-down method, middle point subdivision. Each frame the BVH is traversed and lists of visible primitives are determined, which are then rendered. This project evaluates 3 optimization techniques introduced in [1].

To do so, it was necessary to create an application, which can load .obj scenes and contains a user-controlled camera. The user can also toggle the optimizations in real time. The application displays various statistics about the scene and the culling algorithm. It also supports recording and playback of camera flythroughs.

==

2) used literature

[1] Assarsson, Ulf, and Tomas Moller. ”Optimized view frustum culling algorithms for bounding boxes.” Journal of graphics tools 5.1 (2000): 9-22.
[2] Gribb, Gil, and Klaus Hartmann. ”Fast extraction of viewing frustum planes from the world-view-projection matrix, 2001.” URL http://www.cs.otago.ac.nz/postgrads/alexis/planeExtraction. pdf (2004).
[3] Pharr, Matt, Wenzel Jakob, and Greg Humphreys ”Physically Based Rendering: From Theory To Implementation” [online]

==

3) application command line arguments
Note that only triangle-only scenes are supported!

```
-s sceneName (required)
```

Camera options

```
-vp xCamPos yCamPos zCamPos
-vd xCamDir yCamDir zCamDir
-vu xCamUp yCamUp zCamUp
-vf fov ([radians])
```

Camera movement options

```
-t playback_start_t [from interval <0.,1.>] (playback is paused by default after
launch)
-p playback_file_name
-r rec_out_playback_file_name
-u uniform_playback_t_step_size (useful for measuring stats to ensure that camera
```

views are the same every run)

Statistics export
-m stats_out_file_name
-q (write stats and quit - either after playback if -p is specified, or after one second)

Frustum culling options
-c max_primitives_in_leaf_count
-no-frustum-culling
-no-octant-test
-no-plane-masking
-no-plane-coherency
-no-camera-coherency

==

4) application controls

r ... append node (current view) to camera route - used for recording camera flythrough route
p ... toggle camera playback pause
+ ... increase camera speed (*2)
- ... decrease camera speed (/2)

b ... toggle back face culling
f ... toggle frustum culling
o ... toggle octant test
m ... toggle plane masking
l ... toggle plane coherency
c ... toggle camera coherency (reuse culling result if view has not changed)

==

5) supported platform and build dependencies

The application was tested (and can be compiled) on Linux using gcc 6.3.0 and on MS Windows 10 64-bit using MSVC 2017.
CMake is used to generate build system configuration. On Windows there were troubles with glut include path so it had to added manually to the generated solution.

dependencies:
CMake version 3.2
OpenGL 4.5 (It could probably be easily made to work on lower versions.)
GLUT
GLEW (tested with version 2.0.0-3)
glm (tested with version 0.9.8.3-3)