Programming Project BSP Trees:

Implementing Near Real-Time Shadow Generation Using BSP Trees

Group 2: Yurun (Fox) Chen, Nate Grandner, Aykut Simsek

05/13/2013

1. Introduction

This paper contains the documentation of our 2nd programming project for CS 6703 Computational Geometry Class. The project deals with constructing BSP Trees for static polygonal environments and extends the regular BSP Tree implementation with shadow generation using BSP Trees with algorithm described by Chin and Feiner [1]. In adition to original BSP Tree of the polygons defining the scene, the program creates SVBSP (Shadow Volume BSP) Trees for each light source representing the shadow volume of the polygons facing it. The results are displayed in the graphical interface, and we can see the algorithm runs efficiently by moving the objects and lightsources around the scene.

2. Implementation

The project was implemented in C++ language. We used OpenGL to render the scene from created BSP Tree structures. We divided the implementation into 3 iterative steps where each of us were responsible for one of the iterations.

Part 1:

Fox implemented the initial skeleton of the project, including the main data structures like BSPNode, BSPTree, Polygon and Utility functions required for operations in the algorithm.

Part 2:

Aykut has extended the BSPTree structure to Shadow Volume BSP, by implementing the algorithm defined in the paper and created PointLightSource structure for illumination.

Part 3:

Nate was mainly responsible for implementing the Graphical User Interfaces of the project. During his work along with the GUI, he also applied made modifications and optimizations on the current implementation to speed up the algorithm which made it scalable for scenes containing large number of polygons in it.

3. Usage

- Compile the source code with make command.
- To run the program use the following command with compiled source. You can use the object files submitted with the project for testing.

BSP <scene file> [<obj file> [<scale> [<translatex> <translatey> <translatez>]]]

Some examples runs;

BSP simple.scene cube.obj

BSP simple.scene teapot.obj 0.01
BSP simple.scene skyscraper.obj 0.01
BSP simple.scene shuttle.obj 0.1

- To control the running program;

  Keyboard:

  w,a,s,d,q,e: Move current light

  -,=: Zoom in and out

  0-9: Select light

  l: Enable/disable lighting

  o: Enable/disable outlines

  y: Toggle single light mode

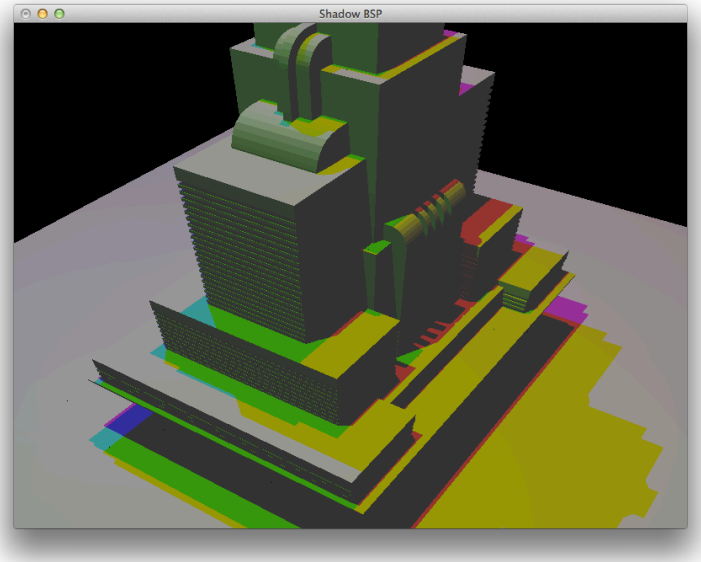
  f: Switch traversal direction for BSP (to show that we're using Painters' and not using a depth buffer)

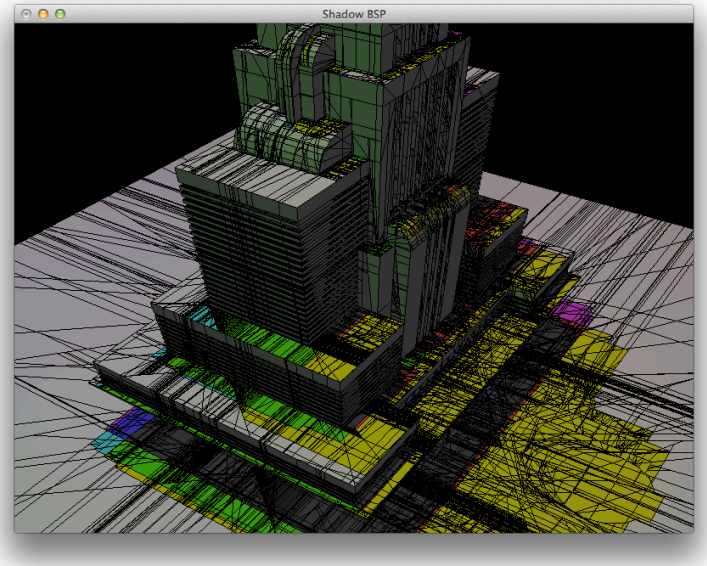
  esc: exit

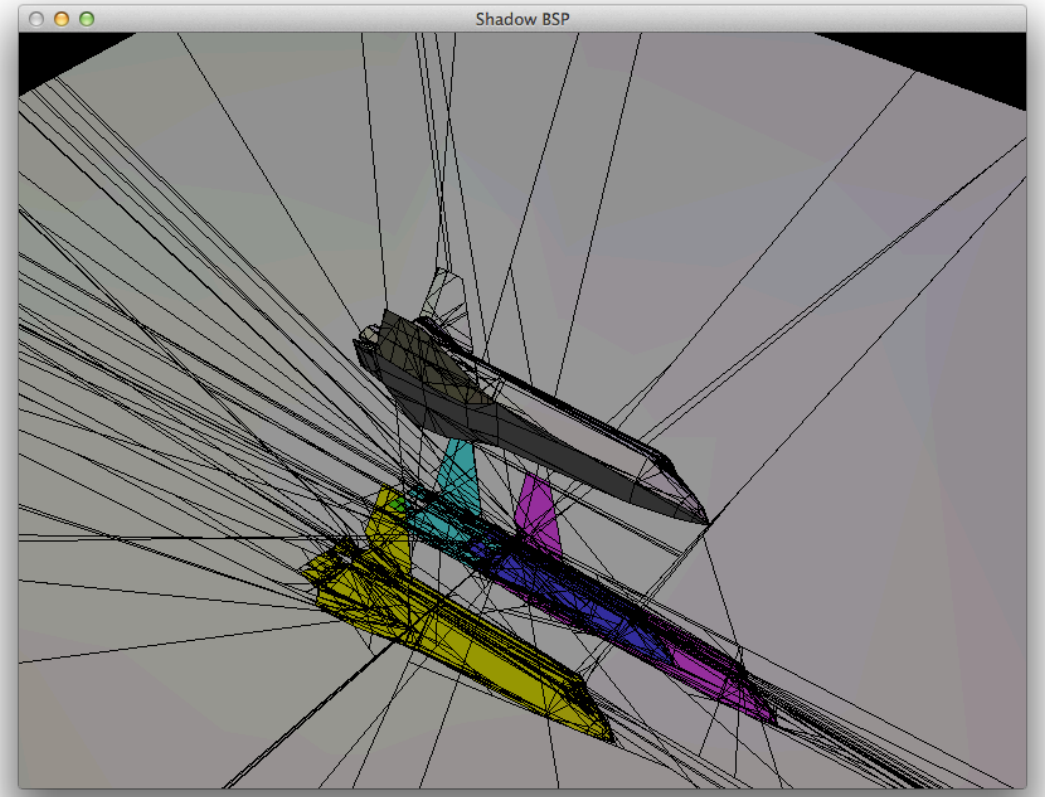
  Mouse:

  Click and drag to look around. Currently its always pointed at the origin.

4. Screenshots







5. References

[1] Chin, Norman, and Feiner Steven. "Near Real-time Shadow Generation Using BSP Trees." *Computer Graphics, Volume 23, Number 3* (1989): 99-106. *Near Real-time Shadow Generation Using BSP Trees*. Web. 27 Apr. 2013.