CEIG – Spanish Computer Graphics Conference (2012), pp. 1-4
Isabel Navazo and Gustavo Patow (Editors)

A GPU BasedMethod for the Automatic Generation of Near-Optimal Navigation Meshes

R. Oliva, N. Pelechano

Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract

In this paper we present a novel, robust and efficient GPU based technique to automatically generate a Navigation Mesh for complex 3D scenes. Our method consists of two steps: firstly, starting with a 3D scene representing a complex environment of one floor with slopes, steps, and other obstacles, it automatically generates a 2D representation based on a single polygon (floor) with holes (obstacles).This step can handle degeneracies of the starting 3D scene model, such as interpenetrating geometry. Secondly, a novel method that exploits the GPU efficiency is used to automatically generate a near-optimal convex decomposition which will represent the cell and portal graph of the environment. Such convex decomposition is a 2D representation of the walkable areas of the environment with portals indicating the crossing borders. The results show that the presented technique not only is more robust than previous CPU methods, but also for the tested environments with up to 1000 vertices is five times faster.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computational Geometry and Object Modeling]: Geometric algorithms, languages and systems

1. Introduction

A popular solution to solve the problem of navigation in a complex scene, consist of subdividing the scene into convex regions (cells) forming what is commonly known as a *Navigation Mesh* (*NavMesh*). A Cell-and-Portal Graph (CPG) is then created where a node of the graph corresponds to a convex region of the *NavMesh* and a portal is an edge shared by two cells. Path-finding can then be solved using an algorithm such as A*.

Although *NavMeshes* are widely used on complex applications such as videogames and virtual simulations, there are not many applications to automatically generate a *NavMesh* appropriate for path planning, so often either the user need to refine those semi-automatic *NavMeshes*, or create them by hand from scratch which is extremely time consuming and a source of errors.

In [OP11], we presented a method to automatically generate near-optimal *NavMeshes* in the CPU. The method, entitled ANavMG, calculates for every notch (concave vertex) of the scene, the closest element that lies on the area formed by the prolongation of the edges incident to the notch, and creates a portal between the notch and the closest element, which can be a vertex, an edge or a previously created portal. The main target of this algorithm is to create a near-optimal convex decomposition of the scene, and the brute force version of the algorithm solves the problem in quadratic time.

In this paper we present an efficient and robust GPU approach to speed up the step of searching for the closest element, based on performing renders of the scene for each notch with the camera parameters defined by characteristics of the notch. The new method not only has higher performance values which allow the user to have very large environments, but it is also more robust since it overcomes several problems that ANavMG presented. Unlike our previous work which created *NavMeshes* for 2D floor plans, this new method can also deal with complex one-level 3D environments.

The main contribution of this paper is a novel GPU based method to generate a *NavMesh* for a given 3D scene, representing a complex environment of a single floor. Our method has two main steps: firstly it abstracts away the information of the 3D model that represents the scene (with its slopes, steps and other obstacles) to automatically convert it into a 2D representation based on a single simple polygon (floor) that can contain holes (obstacles). Secondly, it automatically generates a near-optimal convex decomposition of this 2D representation. Our method is robust against degeneracies of the starting 3D model, such as interpenetrating geometry.