The Method

A Method for Interactive Procedural Generation of Worlds using Dynamic Floor plans / Surfaces

Main problem statement: Does this alternative 'organic' modeling framework work?

sub-problem statements:

- *how do we represent the growing surfaces?*
  - **how does the user control them?*
  - **how does the AI control them?*
- *How does the growing surface method compare to existing methods.*
- *Is there a way we can seed growing surfaces to create meaningfull scenes in terms of composition?*
- *Does the interaction between growing surfaces + governing AI lead to a good automatic modeling system. (define 'good')*

1. **Motivation**

This thesis hopes to contribute to the field of virtual world modeling by combining procedural modeling techniques with interactive intuitive user control schemes. A solely procedural solution, with input parameters and rule based systems as the only user interaction with the system, has the advantage of being highly time efficient but has the obvious downfall of limited control. Conventional modeling tools provide large tool sets which provide users with complete control on geometric shapes but are time consuming. It is interesting to look at the work of Muller et al with respect to (real-time) user control. Their work on procedural generation of urban areas has greatly contributed to the field. Muller et al. describe a complete procedural system for modeling of urban areas based on l-systems and shape grammar. Geometrically, a city basically consists of a street network and buildings. Since a street network is generally very large there is a need for tools which control the geometric properties of the network in an efficient way. In <Interactive Procedural Street Modeling by Guoning Chen> they present such tools using a underlying tensor field which locally and globally controls the direction of streets. The tensor field provides intuitive interaction control with the network and allows for brush techniques and local customizations.

2. Overview

The modeling framework that is proposed in this thesis is aimed at efficient generation of geometry from 'polygonal floor plans'. The representations of these so called floor plans in this thesis are simple polygons in the plane. This paper's main contribution is describing the novel method of generating these floor plans. This thesis is also about the balance between interactivity by user control and efficiency by procedural generation in procedural modeling tools for urban areas and other types of terrain. This thesis discusses and utilizes techniques from geometric and biological modeling for the procedural generation of 2D environments, with an emphasize on game environments. The

techniques used in this paper are greatly influenced by the field of procedural city generation and the more mature field of procedural generation of plants and trees.In order to demonstrate some of the ideas of real time user control on procedural methods discussed in this paper I have developed a simple pseudo procedural modeling tool. The main entities in this tool are growing surfaces which can be manipulated by the user in various ways, but also move on their own acting on the rules which have been given to them. The novelty in this approach lies in the fact that the vertices of the boundary of the growing surface are bodies in a physics simulation. User control over the growing polygons is provided with several handles:

1. A dynamic inner skeleton which is formed by the contour of the polygon.
2. The vertices of the polygon contour can be dragged.
3. Scripting properties of the growing surface.
4. Global properties of the system

The polygonal floorplan has two functions: It can function as container for other polygonal floorplans and it can form the base floorplan for procedurally generated geometry. Procedural generation of geometry for virtual worlds has been a very active research topic for some time now. Prusinkiewicz was one of the first to successfully apply procedural techniques (L-systems) to plant modeling.

3. Growing Surfaces in a Plane: Floorplans for Procedural Geometry

This thesis proposes a novel approach to the creation of polygonal floorplans. This section describes the method and provides motivation for this alternative approach. The initial problem which we stated was the following:

"How can we generate a 3D virtual scene from a set of seed points S in the plane?"

Basically the situation is as follows: we have a 2D bounded plane (or canvas) C and we place (seed) some points p(x,y) in this plane. The goal is to cover the plane with surfaces. By placing initial polygons (which are the geometric representatives of a surface in our approach) on each of the seed points and then expanding them through time, adding vertices as the distance from neighboring vertices crosses a threshold, we come a long way of reaching this goal. Obviously there is a need for higher logic to guide the expansion process, thus we devised several methods for this purpose. A surface has a physical boundary consisting of a configuration of connected physical bodies. These are the rules to handle collisions between a physical body ***b1*** and a physical body ***b2*** which I will address in section 3.3 and 3.4.
The surfaces grow by applying a force directed outwards from the polygon on the physical bodies of the boundary. The directionvector of this force is the same as the direction from a ***local skeleton vertex*** (control point) to the physical body. Now would be a good time to introduce the inner skeleton for our growing surfaces. The skeleton is a vital part of this approach for it provides the control points which locally determine the expansion direction of the physical bodies and provide a handle for the user to manipulate the structure of the surface. I will discuss the skeleton in section 3.2. To add meaning to

the surfaces we introduce properties (parameters) such as expansion speed, surface type (land, building, forest, etc), event types. To enable a large degree of flexibility, properties of surfaces are retrieved from the scripting system. Events are also partly handled by the scripting system. In the next section I will further explain the boundary implementation for the growing surfaces.

3.1 Boundary Implementation.

We identify two possible types of implementation for the surface boundary. The boundary can exist of closely placed physical cells. The cells are 2D objects wisth some simple basic shape such as
a circle, a square or a hexagon. The second way we can implement the boundary is with physical edges, the edge will need to be able to stretch, requiring the physical bodies to be transformed frequently. To decide which method is best we have to examine the efficiency of both methods. The frame rate of the application must allow for realtime user interaction.

Additional topics

To what degree does the physical simulation system determine the translations of physical bodies and when does the rule system dictate the translations of the physical boundary of a surface?

3.2 The Inner Skeleton: Transforming the Straight Skeleton Method to enable an Intuitive Control Scheme.

The skeleton of a polygon is used as a shape descriptor in media retrieval applications.

3.3 Properties of Growing Surfaces.

We identify the following groups of properties:

- Properties which determine growth and shape of the surface.
- Properties which dictate rules for contained surfaces.
- Properties which determine resulting procedural meshes.
- Properties which determine rules for the interaction between neighboring surfaces.

3.4 Event Handling.

Events are triggered in several situations. When two surfaces collide a collision event is triggered. It depends on the types of both surfaces what action will be taken. The script file of a surface determine the consequences for the respective surface.

4. Procedural Techniques for Generation of Geometry

4.1 Graph-like Structures (roads): L-systems.

L-systems have been successfully applied to plant modeling and geometric road network modeling. We will only provide a small summary of the material since we have not actually used any type of L-system.

4.2 Structured Geometry (buildings): Shape Grammar.

This thesis uses a small set of symbols to be able to construct basic buildings using Shape Grammar.

5. Low level Generation of Geometry

This section covers the methods we employ for the final stage of building a virtual scene in our editor: transforming the floorplans, shape grammar and other rules to 3d meshes.

5.1 Polygon Triangulation.

Polygon triangulation is a fundamental topic in computational geometry. Many types of triangulation algorithms are in existence. We need a triangulation method which minimizes the triangle count.

5.2 Extruding 3D Meshes from the Ground Plane

For the actual generation of 3D building geometry we need to be able to translate the shape grammar to combinations of geometric primitives with prefabricated 3D models such as windows and doors. An additional part of the building geometry, the rooftop, is discussed in section 5.3. The generation of elevated terrain is a simpler problem, we have already defined the triangulation of the polygon in the previous section , now we need a function that can generate wall geometry which connects the extruded polygon to the base surface. In section 7.2 I will describe the extrusion control in further detail.

5.3 Generating Rooftops: re-using the Straight Skeleton.

An interesting use of the straight skeleton is the generation of rooftop geometry using the structure of the straight skeleton.

6. Governing AI

6.1 AI Control of Growing Surfaces: The Problem of Meaningful Seeding Growing Surfaces on a 2D plane.

How do we define a meaningful configuration? After establishing the rules for meaningful configurations how do we implement them.

7. User Interaction Methods

7.1 Interaction with the Floorplan Surfaces

The user is able to interact with the surfaces in several ways. A user can directly alter the configuration of a surface by manipulating the skeleton or the physical bodies defining the boundary. Users are also able to change parameters of a surface, such as grow speed and surface type.

7.2 A Simple Control Scheme for the Creation of semi-rule based Geometry: G= f(x) -generated geometry *G* as a result of a single parameter *x.*

The shape and size of a procedurally generated model depends on the z distance the user extrudes the base floorplan from the canvas.