Making Designs Come Alive: Using Physically Based Modeling Techniques in Space Layout Planning

Scott A. Arvin, Donald H. House
Visualization Laboratory, Department of Architecture,
Texas A&M University, College Station, TX. USA

Keywords: physically based space layout, physically based design, responsive design, space layout planning, computer-aided design, human-computer interaction.

Abstract: This paper introduces the concept of responsive design. It elaborates this concept as an approach to free form, adaptable, automated design applying physically based modeling techniques to the design process. Our approach attempts to bridge the gap between totally automated design and the free form brainstorming designers normally employ. We do this by automating the initial placement and sizing of design elements, with an interactive engine that appears alive and highly responsive. We present a method for applying these techniques to architectural space layout planning, and preliminary implementation details for a prototype system for developing rectangular, two-dimensional, single-story floor plans.

1. INTRODUCTION

During schematic building design, an architect develops an overall aesthetic motivation for the building, while trying to arrange individual spaces so they meet the adjacency and area requirements specified in the functional program. The process is often thought of in terms of sculpture, where a design is molded, formed, and massaged. Architects view a building design as constantly evolving and rearranging while they modify existing designs and discover new ones. There has been much research in the past few decades into ways to automate the design process, but most suffer from a lack of intuitive user interaction and a discontinuity with the design language of architects. Architects typically resist using these tools because they tend to be inflexible, prescribing designs rather than allowing the architect to discover designs.

Now, imagine a drawing of a floor plan that is constantly moving, shifting and changing as the designer works on it. If a room is moved out of the building envelope, the building closes in and repairs the hole. If the room is replaced in a different location, the rooms around it adjust to make space for the new addition while remaining the size they need to be. The drawing responds to these changes by managing adjacency and area requirements, allowing the architect to concentrate on design. One could call this *responsive design*.

As we envision it, responsive design is a process that allows a designer to easily make decisions whose consequences immediately propogate throughout the design. Such a responsive design process would be automated, natural, intuitive, flexible, and interactive. It would be automated in that given a set of design intentions, a design solution can be produced. It would be natural in that design elements and intentions are specified in the working language of the designer rather than a low-level representation. It would be intuitive in that the individual elements and the overall design act in expected ways. It would be flexible in that designs are easily modifiable and the design process does not specify the “best” design, but enables designs to emerge. Finally, it would be interactive in that responses to user inputs happen in real time.

We are attempting to create a methodology for a responsive design process by applying physically based techniques to architectural space planning. Although we believe that responsive design can be applied to many graphic design domains, space planning is a relatively simple and well-studied area in which to begin to understand and apply the complexities of this methodology.

In our approach the architect defines programmatic objectives in the usual manner, which are then modeled as physical objects and forces used in a dynamic physical simulation. We model spaces as masses, with adjacencies between spaces modeled as springs connecting the masses. Objectives specified in the architectural program are translated into forces applied to the masses. A dynamic simulation proceeds allowing the mass-spring system to quickly reach equilibrium. The designer then modifies and adds objectives by directly manipulating the graphic model rather than by re-specifying design objectives in the language of the underlying system. The mass-spring representation allows the graphic model to immediately adapt to those changes. It is important to note that we do not intend to simulate the actual behavior of building elements, but to simulate the way architect’s may view and interact with design elements during their conception. One way to think about this approach is that it simulates the design behavior of voids rather than the actual behavior of solids.

It can be argued that given enough computing resources, it may be possible for most design methods to be made responsive. The mathematical and computational requirements in physically based modeling are much more extensive than those required by 3D graphics, so why use this approach? First, physical laws are ingrained in all of us, whether or not we know the mathematics behind them. Our minds and bodies know the physics of structure and motion, otherwise we would not be able to survive. Even though we do not think of spaces and walls as being alive and moveable, spaces that move based on design intentions and that act like physical objects should be intuitive to the user. Second, working with a model whose motion is a result of mass-spring interaction should give somewhat the same feeling as working with a deformable substance such as clay. Third, it is conceptually simple from the user’s point of view. Rather than having to define a long list of highly specific knowledge-based rules, the designer can think in terms of where a space *wants* to be, which is translated by the system into a force that moves the space. The underlying dynamic computation is complex, but the conceptual interface can be fairly simple and intuitive. Finally, it provides a means for finding a *locally* optimum design solution.

Our approach combines the advantages of automation and manual flexibility by producing a locally optimum space layout solution from an initial arrangement of spaces, *and then letting the designer design*. An approach that provides a globally optimum solution can be inflexible and may not be as useful to designers.

Preliminary results of using physically based techniques on simple floor plans indicate that this responsive approach is extremely promising. We have to date been focusing our efforts on a pilot study, developing this approach for single story, rectangular spaces. But even on simple models it feels natural and fun to move spaces around to try out many designs. The ease and speed of generating and manipulating new designs allow new and more interesting ideas to emerge that would be more difficult to discover with non-responsive approaches.

2. BACKGROUND

There exists a wide body of research into automated space planning methods. We present here just a few of those methods that contrast with ours. We then describe the concept of physically based modeling and review some of basic components used in physically based simulations.

2.1 Automated Space Planning

Architectural space layout problems tend to be ill-defined (Yoon, 1992, p. 8) and over-constrained. Problems that are not well-defined are ill-defined (Simon, 1973), in that the initial constraints on the problem are not fully formulated. Resolving ill-defined problems is a process of searching for and refining a set of design constraints. Problems that are over-constrained have too many possible solutions (Balachandran and Gero, 1987). Automated space planning systems need a method of providing a good solution from a large set of possible solutions, and a method of allowing the designer to modify the set of design constraints to continuously refine the problem definition.

Some approaches to automated space layout planning use an iterative process of constructive initial placement, in which spaces are positioned one at a time (Liggett and Mitchell, 1981; Flemming and Chien, 1995). In these approaches, an ordering function is needed to determine which space to position first.

Some approaches are generative in nature, in that they seek to produce all or a large number of the possible designs within a design space. Some examples are evolutionary design techniques (Jo and Gero, 1998; Gero and Kazakov, 1998), and shape grammars (Flemming, 1987).

Methods of producing optimal space plans have been the focus of many approaches such as Liggett and Mitchell (1981).

Constraints have been used in architectural design (Gross, 1986) in, for example, three-dimensional solid modeling (Tobin, 1991; Martini, 1995), and space layout planning (Yoon, 1992).

One of the most notable and extensive research projects in recent years is the "Software Environment to support the Early phases in building Design," or SEED (Flemming and Woodbury, 1995). SEED partitions the schematic design problem into a variety of modules, one of which is SEED-Layout (Flemming and Chien, 1995). SEED-Layout supports design space exploration through an iterative, constraint based approach that can be either manual or automated. It also supports a case-based approach where previous designs can be used to produce new designs.

2.2 Physically Based Modeling

Physically based modeling is a subfield of computer graphics and visualization. It attempts to represent dynamic motion and changes in geometry by modeling objects as mechanical elements that behave according to the laws of physics. Dynamics are most often derived by the use of forward numerical simulation over discrete time intervals. In a forward simulation, the system is moved from its state at the current time to its state at the next discrete time step, using forces to determine accelerations, and thus changes in velocity during the time step, and velocities to determine translations. The process of making this forward extrapolation is called numerical integration. An excellent introduction to the concepts of physically based modeling and a practical guide to the implementation of these concepts in the computer is given in Witkin and Baraff (1997).

Physically based modeling has been used to model the realistic behavior of rigid bodies (Barzel and Barr, 1988; Baraff, 1989) as well as deformable models (Terzopoulos et. al., 1987). Others have recently begun to use dynamics in geometric design. Qin and Vemuri (1998) and Mandal, et. al. (1997) use physically based techniques to interactively manipulate smooth surfaces of arbitrary topology. In their approach, a user defines the points of an initial control mesh, which are manipulated by applying synthesized forces until the desired shape is achieved.

Harada et. al. (1995) use a physically based approach to allow for discrete and continuous manipulation of generalized floor planning problems. They demonstrate their approach on architectural floor plans represented as rectangular dissections, as well as circuit board layout and page layout.

There are three key concepts in physically based modeling needed to understand the rest of this paper. These are the spring-mass-damper modeling element, and the notions of colliding and resting contact between objects.

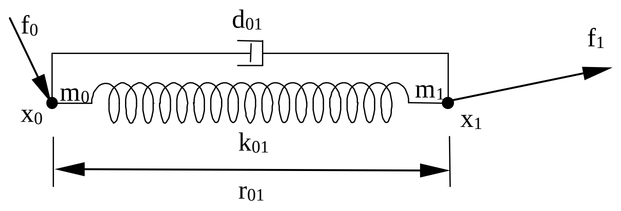


Figure 1. Two masses connected with a spring and a damper.

Figure 1 shows a simple spring-mass-damper system. It consists of two points with mass m_0 and m_1 connected by a spring with spring constant k_{01} and a dashpot with damping constant d_{01}. The spring exerts forces on the two masses with magnitude proportional (with proportionality constant k_{01}) to the difference between the rest length r_{01} of the spring and its current length. The direction of the force will be along the line connecting the point masses. As the masses move farther from each other, the spring applies a force to try to move them closer, and as they move closer the spring tries to separate them. The dashpot is attached in parallel with the spring, and acts like the hydraulic piston on a screen door closer. It damps the motion of the masses by producing forces proportional (with proportionality constant d_{01}) to their relative velocity towards or away from each other, thus reducing the kinetic energy introduced by the spring forces.

Colliding contact occurs between two objects at that instant in time when they touch each other, and have a non-zero relative velocity towards each other. *Collision detection* is the process of determining if and when two objects collide. *Collision response* is the process of determining the result of a collision. (Moore and Wilhelms, 1988) Two bodies are in *resting contact* if they are touching each other, the relative velocity between the two bodies is zero, and the relative acceleration of the bodies is zero. (Witkin and Baraff, 1997)

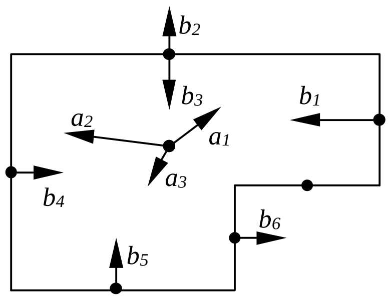
3. PROBLEM

Our problem is how to create a responsive design system within the domain of architectural space planning. Space layout planning can itself be broken down into two problems, determining topological or qualitative properties, and determining geometric or quantitative properties (Jo and Gero, 1998; Flemming, 1989). The topological problem is one of determining the relationship between individual spaces, without regard to the dimensions of any building elements. The geometric problem is one of determining the physical dimensions of the building elements. Both of these problems need to be solved in any approach to space planning.

Our solution to this problem arises out of taking a physically based viewpoint. We treat each space as a solid with mass. Design objectives are treated as forces that are applied to these spaces, and collision detection is used to keep the spaces from overlapping. One of the primary advantages of using a physically based approach is that it is possible for all design objectives to have an effect on the final location of plan elements.

4. IMPLEMENTATION

In applying physically based techniques to space planning, the first problem is how to represent the elements of a space plan such that forces can be made to act on them. The second problem is how to represent architectural design objectives as forces that can be applied to these elements. It is useful to think of design objectives as wants or needs. For example, space A “wants” to be next to space B, or space C “needs” to be 200 square feet in area. It then becomes easier to understand and define the physical forces needed to accomplish the design objectives. *Figure 2* shows a single space with forces acting on its elements. Arrows labeled with *a* represent forces applied to the space location, and that may change the way this space relates to another. Arrows labeled with *b* represent forces applied to the polygonal edges of the space boundary, and that may change the geometric position of the edges.



a: Forces applied to the space location.

b: Forces applied to the polygonal edges of the space boundary.

Figure 2. Simple space with forces acting on its elements.

In this section we present the details for implementing a physically based space planning system. We discuss how to represent spaces with polygonal boundaries and nodes that can be used in a dynamic simulation, how to represent design objectives as forces that can be applied to nodes, and the process for resolving a space plan from a given set of design objectives.

4.1 Node types

A node is a point in space on which a force can be applied. The data structure representing a node in our implementation contains values for mass, position, and velocity, as well as a force accumulator and other geometric information that may be required for each node type. Each unique node type has its own graphic representation. Nodes are typically connected to other nodes by springs. The type of the node determines how its movement, and the movement of the node to which it is connected, is constrained. The two types of nodes we use are the point node and the line node.

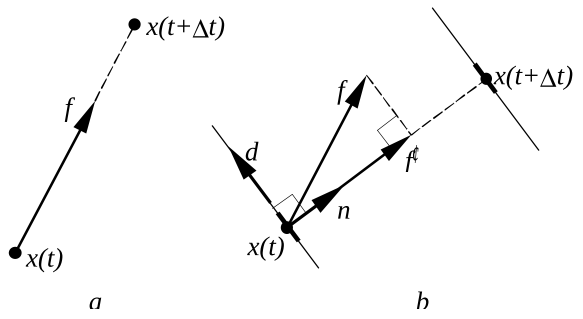


Figure 3. Node types.

Point Node. A point node is the simplest type. The data structure for a point node stores typical node information, and it is displayed as a dot. A force applied to it is not constrained in any way. *Figure 3a* shows a point node with position *x(t)* at time *t*, with a constant force *f* applied to it, and no initial velocity. *x(t)* is accelerated in the direction of *f* so that at time *t*+Δ*t* it is at position *x(t*+Δ*t)*.

In our implementation we use point nodes to define the center of spaces.

Line Node. A line node defines an infinite line passing through a point. Any forces applied to a line node are constrained to act perpendicular to the line it defines, thus preserving its orientation. The data structure for each line node contains unit direction and unit normal vectors in addition to the typical node information. A position and a direction are all that are needed to define a line; the additional normal vector is stored so as not to repeat its calculation. A line node is displayed as a dot with a short bar going through it parallel to the direction vector. *Figure 3b* shows a line node with unit direction *d*, unit normal *n*, and position *x(t)* at time *t*, with a constant force *f* applied to it, and no initial velocity. *f* is constrained to act along *n* by

$$f' = (f \cdot n)n,$$

where the · operator is the dot product of two vectors. *x(t)* is accelerated in the direction of *n* so that at time *t*+Δ*t* it is at *x(t*+Δ*t)*.

In our implementation we use line nodes to define the polygonal edges of space boundaries. They could also be used to define linear spaces such as corridors.

4.2 Polygonal shapes

Shapes are limited to polygons with non-crossing edges and are defined as an ordered list of line nodes, each connected to a common center node. *Figure 4* shows an arbitrary *n*-sided polygon, with center node *c*, edge line nodes *e*, and vertices *v*. Each vertex v_i can be found as the intersection of the direction vectors of e_i and $e_{i\oplus 1}$, where the operator ⊕ is addition modulo *n*.

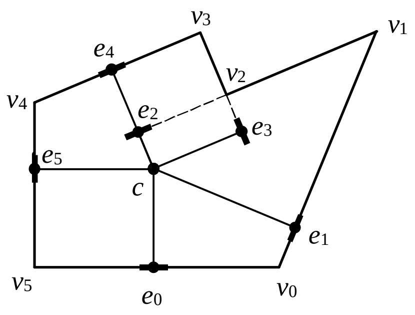


Figure 4. An arbitrary polygonal shape represented with edge line nodes.

Although this may seem like a complicated way to represent a shape, when used to define the boundary of a space it allows us to apply forces to individual segments without having to worry about maintaining their orientation. The typical task in space planning is to move a wall, not to move the end points of a wall. If a shape were represented with point nodes at its vertices, in order to maintian orientation any force applied to one vertex would have to be separated into components that are applied to its surrounding vertices. For non-orthogonal shapes, computing the necessary components could get unnecessarily complicated. *Figure 5a* shows a rectangle represented with vertex point nodes. A force applied to one of its vertices as shown results in the non-orthogonal polygon shown in *Figure 5b*. *Figure 5c* shows a rectangle represented with edge line nodes. Forces f_1 and f_2 applied to its edges are constrained to yield f_1¢ and f_2¢, which act normal to each edge, resulting in the maintained rectangular shape shown in *Figure 5d*.

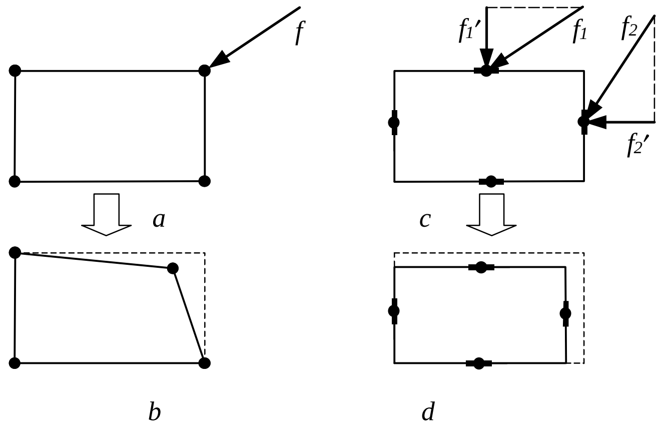


Figure 5. A rectangle represented with vertex point nodes and edge line nodes.

4.3 Spaces

A space defines any arbitrary area or volume. The data structure representing a space contains a common center node to define its location, and a polygon to define its boundary. These values are optional, to be able to define any generalized space. For example, a space used to represent *the outside* would not need a defined position or shape, and could be used when a building space needs to relate to the outside.

A space may also contain any number of child spaces. Child spaces may, in turn, contain their own set of child spaces. In this way, a hierarchy of spaces of arbitrary depth can be defined, similar to that described by Flemming and Chien (1995). A parent space and its child spaces defines a self-contained physical system, and the relationship between the parent and its children is defined by the parent boundary. If a parent boundary exists, as in *Figure 6a*, the system of child spaces needs to be contained within that boundary. If a parent boundary does not exist, as in *Figure 6b*, the solution of the system of child spaces will define the parental boundary. This spatial structure allows the definition of a wide variety of spaces, and allows for spatial hierarchies of arbitrary depth.

The location of the center node relative to the edges is not important to the polygonal representation, but is important in determining space adjacencies. For simple shapes the geometric center or center of mass is fine, but for more complicated shapes a more appropriate center may need to be defined by the designer.

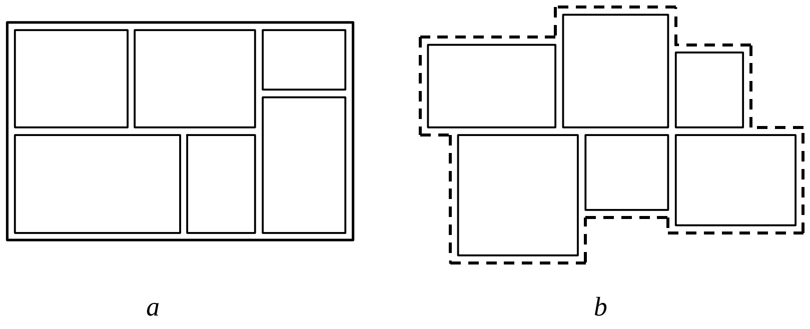


Figure 6. Parent space with and without a defined boundary.

4.4 Objectives

We define a design *objective* as something that a designer wants to happen, specifically, any intention that influences geometry. It could be argued that any intention that determines the position of any object could be called an objective, but for the purposes of our approach we limit objectives to those intentions that affect the position of one space relative to another or that affect the location of walls.

We use the term objective instead of *constraint* to avoid confusion with the term used in physically based constrained dynamics. In constrained dynamics, a constraint is a condition that, once met, continues to be met throughout a dynamic simulation (Barzel and Barr, 1988; Witkin and Kass, 1989). One example is an area constraint, where any applied forces that may act to change the area of a polygon will be counteracted to ensure that the area remains the same. Constrained dynamics is not a process of *constraint satisfaction*, but one of *constraint maintenance*. If an area is not as it should be, constrained dynamics will *not* act to change it to the correct area.

Design objectives need to be translated into forces that act on the nodes. They are separated into two categories, topological objectives and geometric objectives, in keeping with the two problems in space planning noted earlier. Topological objectives apply forces to the center of a space, and geometric objectives apply forces to the polygonal edges of space boundaries. Any design objective that affects the position of any element in an architectural plan can probably be translated into force objectives. Here, we will only give details for the few that we found to be necessary to demonstrate the concept of responsive physically based space planning.

Adjacency Objective. An Adjacency Objective is a type of topological objective. It connects the center nodes of two spaces with a spring, and applies forces to those nodes depending upon the distance between them. Using circular spaces as an example, the rest length of the spring may be the sum of the radii of each space. If the spaces are not located next to each other, the spring will produce forces on each space node that attempt to move them together. The spring constant defines the strength of the adjacency, and is initially set on a scale of 0.0 to 1.0 relative to other adjacency spring constants. *Table 1* shows how descriptive terms for adjacency relationships might be mapped into numerical values for spring constants.

Table 1. Adjacency Spring Constants

Adjacency requirement	Spring constant
Immediate	1.0
Important	0.7
Convenient	0.3
Unimportant	0.0

Area Objective. An Area Objective is a type of geometric objective. Recall that a space may contain a polygon that defines the space boundary, and that a polygon is defined as an ordered list of line nodes. An area objective acts to maintain the area of the boundary polygon. The center node of the space becomes the center node of the polygon, and a spring connects this node with each edge line node. The rest lengths of the springs are used to maintain the correct area of the boundary polygon and are set to the perpendicular distance from the center node to the line node.

4.5 Dynamic Simulation

Once a set of spaces and objectives has been defined, a dynamic simulation is allowed to run to produce a layout solution. In our current implementation, the initial location of each space's center node is chosen at random. The dynamic simulation itself proceeds in two phases due to the separate tasks of the general space layout problem. First, the relationship between spaces needs to be resolved, and then the position of walls separating the spaces needs to be resolved. In our approach they cannot be solved at the same time. For this reason, the first phase is to run the simulation with only topological objectives being applied. Once a topological simulation has reached equilibrium, the second phase is started, in which geometric objectives are applied. Once a geometric simulation has reached equilibrium, the designer can begin to analyze and interact with the design by modifying existing objectives and adding new ones.

Although this may sound like the two steps are turning objectives on and off, in our current approach all of the objectives are being applied at all times. The only difference between the two modes is in the boundary shape used in collision detection and response, as described below.

4.5.1 Topological resolution

The first phase in solving a space layout is to determine the location of each space relative to all other spaces. Any force acting on space centers will affect the position of the entire space. Using the objectives we have descibed previously, adjacency objectives are the only forces that have an effect. For collision detection, boundary shapes are treated as circles and are able to slide around each other if necessary. If polygonal boundary shapes were used, boundary corners may catch each other and keep one space from being able to move to the other side of another. The dynamic simulation runs until the system is in equilibrium, which is defined as the point in time when all velocities are zero.

Collision detection and response during this phase can be simulated with the use of stiff springs between overlapping spaces. Although this method is inaccurate, in that some spaces may still overlap, it is adequate for determining topological relationships.

4.5.2 Geometric resolution

Once the dynamic simulation for space resolution has reached equilibrium, space boundaries are switched from a circular to a polygonal representation. Collision detection and response then act to keep spaces from overlapping, resulting in an arrangement that is very close to a recognizable building floor plan.

Collision detection and response are much more complicated for geometric resolution than for topological resolution. Overlapping spaces are unacceptable, so accurate collision response is required. Especially important and problematic is dealing with resting contacts. Although it would seem that resting contacts would be easy to deal with, their handling is one of the more difficult problems in physically based modeling (Witkin and Baraff. p. D49).

To simulate resting contacts, forces must be applied to each object that is in contact with another object so that the contact will be maintained during the next time step. Although colliding contacts can be solved in turn for each individual contact, all resting contacts must be solved together. For a body in resting contact with two other bodies, the resting forces need to account for both contacts.

In our approach we use constrained dynamics to maintain resting contacts (Witkin and Baraff, 1997). Once two spaces are determined to be in resting contact, a distance constraint is applied between the parallel edge line nodes of the contiguous boundary edges. As noted in section 4.4, this constraint acts to maintain the distance between the two nodes. Constraint forces are then solved as a complete system.

4.6 User Interaction

Currently, the definition of spaces for a design project is accomplished with a text file. In the future we will implement an interface to interactively input and edit design spaces and objectives.

The method of interacting with objects in a physical simulation is different from that used to edit static graphic objects. In a typical CAD application, a user moves an object by directly changing the position of the object. In a forward physical simulation, the inputs to a system are forces, and the outputs are positions and velocities. Directly changing the position of an object is inconsistent with this notion of simulation.

One method of moving objects in a physical simulation is to attach a temporary zero length spring between the mouse’s cursor position and the object being moved. This spring then applies a force on the object in the direction of the mouse’s position. This approach is not usually satisfying to the user, because the cursor position and the object position are not the same during the editing process.

A second method, and the one that we favor, is to temporarily set the mass of the edited object to infinity, and then directly change the position of the object. All dynamic calculations using mass involve multiplication by the mass inverse, so the value of each object’s mass inverse is stored instead of its mass. To set a mass to infinity, we simply set the mass inverse to zero, making the object unresponsive to any forces acting on it. Thus, while the user is moving a point it cannot be moved by the dynamics. Once the user lets go of the object the mass inverse is set to its previous value and the simulation proceeds as usual.

4.7 Implementation Details

Software development was done on a Silicon Graphics O2 workstation running Irix 6.3, with a 200 MHz R5000 processor, and 192 MB RAM.

Programming was done in object-oriented C++, using OpenGL for the graphics, and the FLTK user-interface toolkit (FLTK, 1999).

Ordinary Differential Equations were solved using Runge-Kutta fourth order numerical integration.

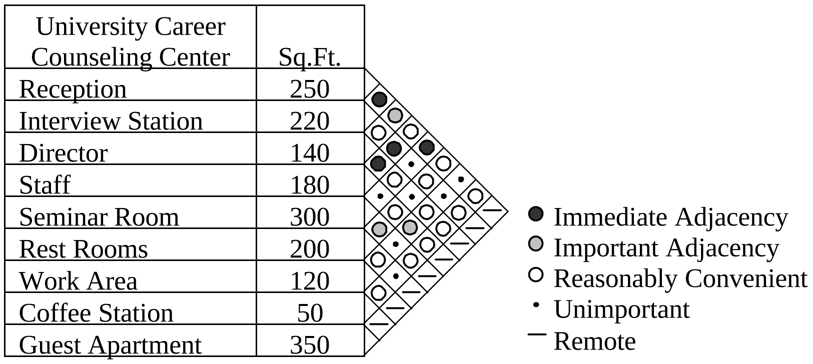
Table 2 lists some of the constants we used in our force computations.

Table 2. Dynamic constants.

Adjacency spring constant	0.0–20.0
Shape spring constant	500.0
Edit spring constant	500.0
Spring dashpot	2.0
Coefficient of Restitution	0.0
Viscosity	10.0

5. RESULTS

During the development of the dynamics for our physically based approach to space planning, we used the architectural program shown in *Figure 7*, from an example by Karlen (1993). During the early stages of development, we needed a program that was fairly small but which contained many adjacency requirements. It needed to be small because the emphasis in early development was in getting the dynamics between individual spaces to work. It needed to have many adjacency requirements so that many locally optimal solutions were possible. A small number of adjacency requirements would yield a small number of solutions.



University Career Counseling Center	Sq.Ft.
Reception	250
Interview Station	220
Director	140
Staff	180
Seminar Room	300
Rest Rooms	200
Work Area	120
Coffee Station	50
Guest Apartment	350

Figure 7. Sample Adjacency Matrix [Redrawn from (Karlen 1993), p. 22]

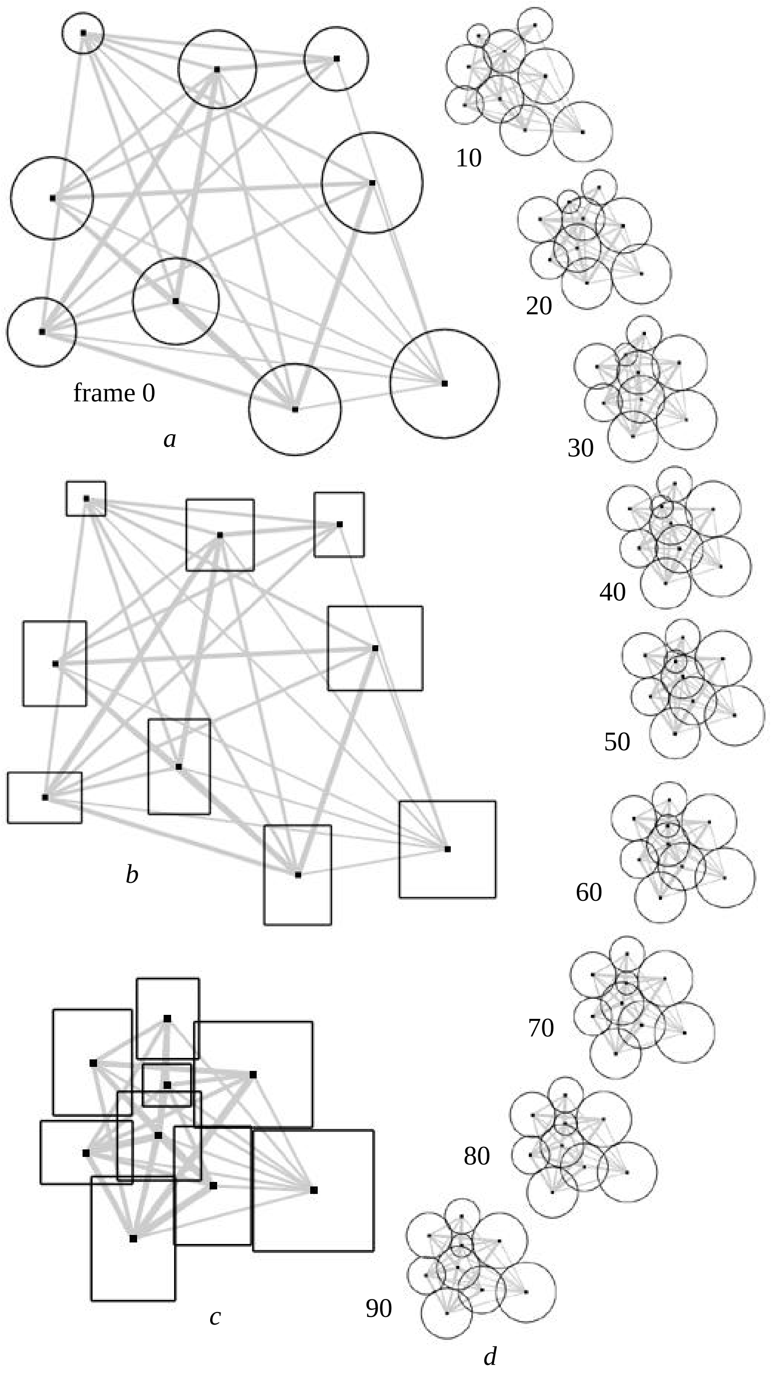


Figure 8. Sample Topological Resolution

Figure 8 shows a sample topological resolution using the program in *Figure 7*. *Figure 8a* and *8b* show each space boundary drawn with its required area and with random initial positions, *Figure 8b* displaying boundaries drawn as rectangles with random proportions, and *Figure 8a* displaying boundaries drawn as circles. Recall that during topological resolution, circles are used in collision detection. It is difficult to show the dynamic movement with static images, but *Figure 8d* shows every tenth frame from the dynamic simulation, with frame 90 showing the spaces in equilibrium. The entire sequence took three seconds to compute and display, so the illusion to the user is of smooth natural motion. Notice that most of the movement occurs between frames 0 and 10 when the spaces are coming together, and that any movement after that is a result of the spaces rearranging themselves and coming to equilibrium. With some initial positions it is possible for the system to almost be at equilibrium when one space manages to move onto the other side of another, and the whole system rearranges itself. *Figure 8c* shows the final topological solution with the boundaries drawn as rectangles again. Although some of the boundaries overlap, this in not important during topological resolution and the overlaps will be resolved during geometric resolution.

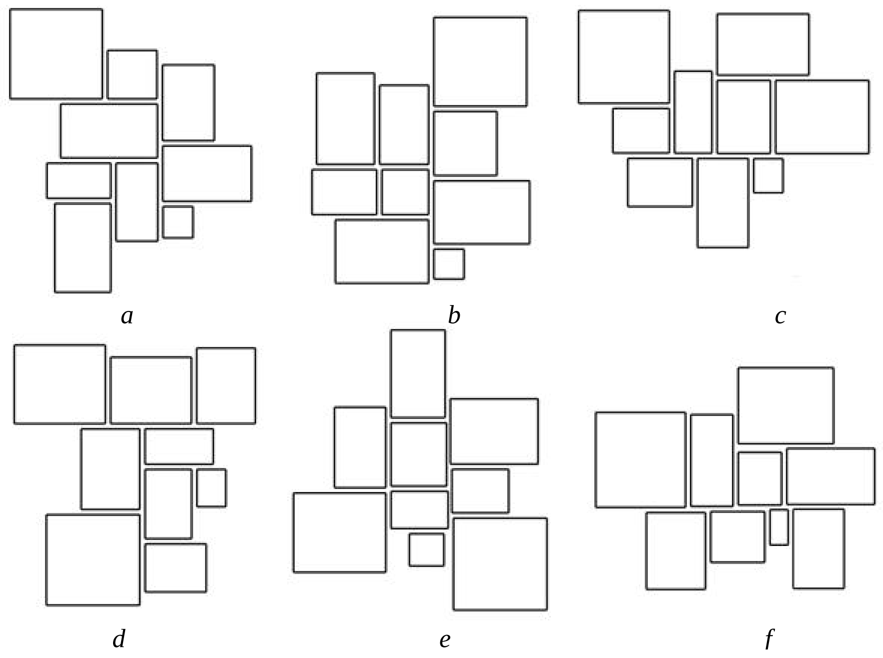


Figure 9. Sample Geometric Resolutions

Figure 9 shows six samples of geometric resolutions using the same architectural program. Initial proportions for each space were maintained from sample to sample, but initial positions were randomized. Final topological relationships were not edited manually. For these results, topological resolution was *not* performed before geometric resolution, so these do not represent locally optimal topological solutions. Note in *Figure 9f* that some wasted space is possible. This problem can be overcome by finding the shortest distance between parallel lines across the opening that needs to be closed, and attaching a spring between the line nodes for those lines.

Note the variety of designs produced from a simple set of objectives. The only objectives active in producing these samples are adjacency and rectangular area objectives. The addition of other objectives such as non-rectangular shape, parental shape, and alignment objectives, among others, should allow the architect to have a great amount of control over the design of space plans.

6. DISCUSSION

We feel that our prototype system provides a convincing demonstration of the attractiveness of the responsive design approach and the use of physically based methods in implementing this approach. It is difficult to convey, in a paper, the experience of working with our system. But, it really does evoke the feeling that one is working with a "living" design, one that responds to the user in ways consistent with programmatic objectives while still providing a high degree of intuitive designer control. We are surprised and pleased at the variety of designs that present themselves from random initial positions. Even when the number of spaces is small (3 or 4), it is enjoyable to rearrange them to see what happens. The fact that the physically based objectives do most of the work allows one to easily explore many design possibilities.

We proposed earlier that a responsive design process would be automated, natural, interactive, intuitive, and flexible. We feel that using a physically based approach to creating a responsive design system meets these criteria very well. Our results show that our approach is clearly automated and flexible. Design solutions are created automatically from a small set of design objectives, and the system is very flexible in that design objects can be easily manipulated, allowing new designs to emmerge. Our approach is for the most part intuitive in that design elements act in expected ways, although some of the springyness of objects may be a little unexpected. Our current implementation is highly interactive and results are produced in real time, but whether or not this will hold for more complicated designs is an open question. The last criterion for responsive design is that designers work with elements and intentions in their natural design language rather than through a low-level representation. How our approach fits this criterion is also an open question. An interactive interface that enables designers to specify objectives directly on the graphic elements of a space plan should provide for natural interaction. However, requiring the designer to have an extensive understanding of the underlying physical model will hinder the feeling of natural interaction.

Within our prototype system, the quality of interaction when rearranging spaces is, for the most part, as we envisioned it to be. When editing a space plan, the designer is able to move spaces around easily to quickly try out new designs. Because of the use of springs and forces, the spaces are very springy feeling. We feel that this can be attractive, in that some springyness adds to the feeling that the plan is alive, but that when it is too noticable it interfers with the interaction. This problem is, however, easily solved by the use of appropriate spring and damping constants.

The responsive feeling of our system depends upon the rate at which calculations can be done and the display updated. The *frame rate* is the rate at which one compute and display cycle is completed. During the final stages of geometric resolution, this frame rate can get a little slow, sometimes approaching non-interactive rates. This is due to two factors. First, accurately detecting and responding to collisions requires that the simulation clock be stopped at the time of collision and the response calculated. Finding the exact time of colliding contact requires a binary search, each stage of which requires a solution to the system's dynamic equations. Second, the handling of resting contacts requires the calculation of constraint forces at each resting contact, which requires the solution of a large set of simultaneous equations describing the constraint dynamics. However, although each of these problems is computationally intensive, their time complexity is at worst a low order polynomial. This makes us hopeful that increasing computer speeds, along with the application of appropriate approximation methods will allow us to achive responsive performance, even for complex building designs, in the reasonably near future.

It is in the nature of the space planning problem is that it is NP-complete, but we do not claim to be able to produce solutions that are globally optimal. Instead, we look for local optima and depend on the designer to recognize when a solution is weak and to make appropriate changes by hand to guide the system into a more optimal configuration. We also feel that we can get some reduction in time complexity by the use of a hierarchical approach. When using hierarchical spaces, such as those discussed in section 4.3, solving for the location of a space is dependent only on the parent and sibling spaces, and is not dependent on every other space in the hierarchy, thus reducing the global problem into a set of smaller, simpler problems.

7. FUTURE WORK

Since the responsive design approach to doing space planning is a previously unexplored concept, it raises many new questions and presents many opportunities for future work. Some of these are obvious and require answers and elaboration in order to make this approach truly useful to space planners. Other questions are of a more fundamental nature.

In the immediate short term, we plan to explore:

– What other kinds of design intentions can be translated into physically based design objectives? Some examples may be daylighting, view, non-rectangular shapes, etc.
– How can this approach be modified to handle circulation? Might the use of line nodes to define corridor space "centers" be a promising solution?
– How can this approach be extended to three dimensions in order to handle multi-story plans?
– What are appropriate values to set for the spring constants? Is it possible to find a range of values that apply to all design situations, or will the designer have to know how to set their own values?
– What are the interface issues in inputing and manipulating design elements?
– What is an effective way of displaying the various forces on a space so that they are meaningful to designers and allow them to understand the conflicting nature of a set of design objectives?

To validate and more firmly ground our work we plan to explore the following fundamental questions:

– Does our physically based approach truly fit the intended characteristics of responsive design?
– How does this approach fit into the architectural programming process?
– How does this approach fit into the overall architectural design process, and how can it be integrated with other aspects of schematic design and design development?
– How does this approach affect the quality of designs that a designer produces? Is it better than existing methods in allowing designers to discover new designs? Will it aid designers from getting fixated on a single design?
– Although this approach is not intended to produce globally optimum solutions, it should benefit the designer to be able to start from a more optimal solution than those we are able to currently provide. How can optimization methods be applied to this approach?
– How much of the underlying physical model should be revealed to designers? Do they need to know about the underlying forces and the details of the physically based system, or is it better for them to understand how it works through the use of another metaphor?
– How can this approach be applied to other design domains?
– What other benefits are there in using a physically based approach? The different forces applied by each objective may be used as a measure of the effectiveness of a particular design, to be compared to other design schemes.

8. CONCLUSION

We have introduced the concept of *responsive design*, as a new paradigm within which tools for architectural design may be developed. We have also demonstrated, with a simple prototype system, that the concepts of physically based modeling provide concrete principles around which to construct such tools. Our preliminary results with the prototype indicate that useful physically based responsive design tools can be built, and that these tools will be practical, enjoyable to use and integrate well with the exploratory nature of the design development process.

ACKNOWLEDGEMENTS

This work was supported in part by an O. N. Mitchell, Jr. Endowed Fellowship in Construction Management, the Visualization Laboratory and the Department of Architecture in the College of Architecture Texas A&M University.

REFERENCES

Balachandran M., and Gero J. S., 1987, “Dimensioning of architectural floor plans under conflicting objectives”, *Environment and Planning B*, 14:29-37.

Baraff D., 1989, “Analytical methods for dynamic simulation of non-penetrating rigid bodies”, *Computer Graphics*, 23(3):223-232.

Barzel R., and Barr A. H., 1988, “A Modeling System Based on Dynamic Constraints”, *Computer Graphics*, 22(4):179-188.

Flemming U., 1987, “The role of shape grammars in the analysis and creation of designs”, *Computability of Design*, edited by Kalay Y. E. (John Wiley & Sons, New York):245-272.

Flemming U., 1989, “More on the representation and generation of loosely packed arrangements of rectangles”, *Environment and Planning B*, 16:327-359.

Flemming U., and Chien S. F., 1995, “Schematic Layout Design in SEED Environment”, Journal of Architectural Engineering, 1(4):162-169.

Flemming U., and Woodbury R., 1995, “Software Environment to Support Early Phases in Building Design (SEED): Overview”, Journal of Architectural Engineering, 1(4):147-152.

FLTK, 1999, *The Fast Light Tool Kit Home Page*, <http://www.fltk.org>, accessed 1 March 1999.

Gero J. S., and Kazakov V. A., 1998, “Evolving design genes in space layout planning problems”, *Artificial Intelligence in Engineering*, 12(3):163-176.

Gross M., Ervin S., Anderson J., and Fleisher, A, 1986, “Designing with constraints”, *Computability of Design*, edited by Kalay Y. E. (John Wiley & Sons, New York):53-68.

Harada M., Witkin A., and Baraff D., 1995, “Interactive Physically-Based Manipulation of Discrete/Continuous Models”, In *Computer Graphics* Proceedings, Proceedings of SIGGRAPH 95, 199-208.

Jo J. H., and Gero J. S., 1998, “Space layout planning using an evolutionary approach”, *Artificial Intelligence in Engineering*, 12(3):149-162.

Karlen M., 1993, *Space Planning Basics* (Van Nostrand Reinhold, New York).

Liggett R. S., and Mitchell W. J., 1981, “Optimal space planning in practice”, Computer-Aided Design

Mandal C., Qin H, and Vemuri B. C., 1997, “Dynamic Smooth Subdivision Surfaces for Data Visualization”, *Proceedings of IEEE Visualization ’97*, 371-377.

Martini K., 1995, “Hierarchical geometric constraints for building design”, *Computer-Aided Design*, 27(3):181-191.

Moore M., and Wilhelms J., 1988, “Collision detection and response for computer animation”, *Computer Graphics*, 22:289-298.

Qin H., and Vemuri B. C., 1998, “Dynamic Catmull-Clark Subdivision Surfaces”, IEEE Transactions on Visualization and Computer Graphics, 4(3):215-229.

Simon H. A., 1973, “The structure of ill-structured problems”, Artificial Intelligence, 4:181-201.

Terzopoulos D., Platt J., Barr A., and Fleischer K., 1987, “Elastically deformable models”, *Computer Graphics*, 21(4).

Tobin K. L., 1991, “Constrain-Based Three-Dimensional Modeling as a Design Tool”, *Reality and Virtual Reality*, 1991 ACADIA Proceedings, edited by G. Goldman and M. Zdepski, 193-209.

Witkin A., and Baraff D., 1997, *Physically Based Modeling: Principles and Practice* (ACM SIGGRAPH 97 Course Notes, Course 19).

Witkin A., and Kass M., 1989, “Spacetime constraints”, *Computer Graphics*, 22:159-168.

Yoon K. B., 1992, *A Constraint Model of Space Planning* (Computational Mechanics Publications, Southampton, UK).