Interactive 3D Modeling in the Inception Phase of Architectural Design

B. de Vries, A.J. Jessurun, J.J. van Wijk
Eindhoven University of Technology, The Netherlands

Abstract

In architectural design 3D modeling in the inception phase is up to now not supported in CAD. Moreover 3D modeling is still far from intuitive with the common interface techniques. This paper introduces a 3D modeling tool for architects, which is limited in its purposes but yet very powerful in its use. A cube is the basic drawing element to construct building masses. Modification of the design is achieved by dragging sets of cubes resulting in the creation or deletion of new cubes.

1. Introduction

CAD tools have developed from drawing tools into design support tools. In the architectural practice though there is still an emphasis on the use as a drawing production tool after the major design decisions have been taken. Computer tools perform very well in editing and in making nice presentations. In the early design stages however, pen and paper and cardboard models are the dominant design tools. Interviews with architects have learned that (1) they prefer direct manipulation of the building masses and spaces and that (2) the shapes should incorporate inaccuracy and vagueness. Design thinking is obstructed by current CAD packages because these two aspects are absent.

In this paper DDDoolz is presented, which is an innovative system dedicated to mass study and spatial design in the early stage of architectural design. Using 3D painting as a metaphor a design system has been developed for creating "rough" three-dimensional solid models with only a few operations.

The outline of the paper is as follows: In the next section a brief overview will be presented of the state of the art in commercial CAD packages and in scientific research. In section 3 the principles of the 3D painting metaphor are explained. Section 4 discusses the system functionality, the implementation and the user interface. The paper is concluded with experiences that are highlighted by examples and with some future directions for DDDoolz.

2. Background

Current commercial CAD packages do support 3D modeling but at the cost of a lot of effort. 3D modeling consists of two basic elements, namely geometrical operations and viewpoint management. In general the operand of a geometrical operation is a geometrical basic primitive (e.g. a cube, sphere, etc.) or a parameterized building element. In the first case (e.g. 3D Studio Max) a stack of operators is used for describing the desired result, which requires quite a lot of knowledge from the user about the operator functionality. In the latter case (e.g. Archicad) parameters like length, width, and height can be manipulated to adjust the result. However, in the early design stage the architect does not articulate his/her thoughts in parameter terms. Since the modeling still takes place on a surface (i.e. the monitor screen) we need extra features to support us in managing the right view on our designed object. Multiple views only partly solve this problem because the user has to preset a viewpoint position and does not really know if the created object will obstruct the view.

In scientific research much effort has been spent on sketching systems that support the design in creating 3D geometries from a 2D drawing plane. Some outstanding examples of these kind of systems are Sketch [10], The Electronic Napkin [6] and Teddy [7]. The starting point for these systems is the line drawing that is interpreted to infer lines that are part of a 3D geometry. The process is executed while drawing to give direct feedback to the user. The viewpoint position is set before performing the drawing operation, which might lead to objects obstructing the view in case of large and complex designs. An electronic stylus is used to stay as close as possible to the paper and pencil sketching method. The geometry construction process is the most critical part, which demands certain training of the designer to make it really productive, or even better, to make it creative.

A different approach that has been hardly developed so far, is direct creation of 3D masses representing architectural entities like walls, floors, etc. in the building environment. The one example that also seeks to provide this functionality is Sculptor [8]. Sculptor's basic principle is the creation of masses and voids by subtracting solids. Consequently, modeling has become very different from creating shapes in a drawing manner.

Supporting architectural modeling by direct manipulation is still far from what it should be when compared with the traditional methods. Though there is much research progress on direct manipulation [3], this is usually in a generic geometric modeling context. Up to now there has been very

little experience in applying direct manipulation in 3D to architectural modeling.

3. 3D Painting

It would seem appropriate to use a 3D input device to assist in this 3D design task. However, our experience was rather disappointing. An experimental application consisting of a virtual pen and ground plane in a 3D environment has been developed for testing. The user has a fixed perspective view on the ground plane and the pen can be moved in any direction. If the pen hits the ground plane a rectangular block is created until the pen is lifted from the ground plane. The height of the block depends on the length of the pen on the upper side of the ground plane. Tests with the experimental application using a Flock of Birds as the 3D input device showed that the pen is technically difficult to control. Precautions in the software have to prevent the pen from becoming very 'shaky'. The hand-eye coordination is uncomfortable, because distance between the sensor and the monitor is often too large to keep control. Finally, force feedback from the drawing board is badly missing and the sensor itself seems a bit awkward if used as a pen.

Apart from pen and paper for sketching, the architect also uses materials like foam that he/she cuts and glues to make mass models on a very abstract level. The physical models are created to give better insight on dimensions, scaling, repetition, symmetry etc.

In the next section we summarize the requirements for a new tool that supports an architect in the inception phase of architectural design. After that, we describe our modeling tool DDDoolz (abbreviated from 3D-Tools). The proposed method indicated as "3D painting", combines the directness of the sketching method with the instant 3D shape representation of the physical model. More information, including the program itself can be found on the web [www.ds.arch.tue.nl/Research/DDDoolz/].

3.1 Requirements

The objectives for the development of DDDoolz were:

- Easy creation and manipulation of masses and spaces

  Creation and ordering of spaces by creating masses that bound them are considered as one of the most important steps of the (early) design process. Spatial dimensions are roughly determined. The mass shape should be easily created and deformable into another shape. Sometimes a shape represents a specific building element (e.g. a window). In that case deformations should be constrained to certain extents.

- Minimal commands

  Preferably no commands at all should be necessary in order not to obstruct the design process.

- Accurate control

  Though the design should be rough, the input device should be very precise.

- Easy visual evaluation of the created model

  Viewing the model from any inside or outside position is required to get feedback on the design.

3.2 Geometric Model

Painting masses with solids in DDDoolz is supported by using a cube as the painting entity. The cube is the building block for larger and more complex shaped masses. For convenience the geometric model will be discussed in 2D, thus reducing the cube to a square.

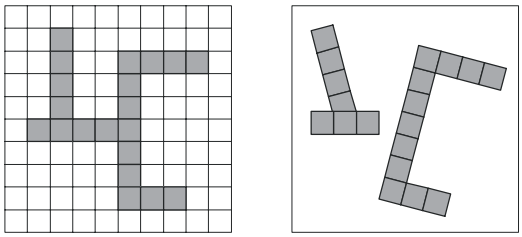

Figure 1: *Grids and Squares*

With squares as the painting unit, the most straightforward approach is using a grid as the geometrical model. The painting algorithm is fast and simple but the resulting masses always will have orthogonal shapes. Reducing the square unit to ultimately the pixel unit will visually solve this problem but then the essence of the application is lost, namely "rough" designing with a high granularity. Therefore we choose for another approach namely, each stroke consisting of a group of squares has its own coordinate system, which follows from the orientation of the starting square. Orientation of the starting square under different angles will give much more freedom in creating shapes.

Although many orthogonal shapes (e.g. walls, floors) are found in architecture, most architects react negatively to being restricted to orthogonal masses. Breaking away from orthogonality, even by a simple feature like rotating the drawing entity, improves the shaping flexibility to a great extent.

In the next paragraphs the geometric model will be extended to 3D.

3.3 Face Orientation Method

Determining the drawing direction in 3D is far from trivial when working with CAAD packages. User coordinate systems are necessary to identify the exact position in space and to construct new elements. For DDDoolz a very straightforward solution was found, called the Face Orientation Method [1]. By picking a face of a cube the drawing plane is defined as the plane that contains this face. Models can be extended from any viewpoint position as long a face can be found within the required drawing plane.

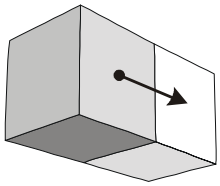

Figure 2: *Face Orientation Method*

3.4 Drag and Copy

Starting from a certain cube, new cubes will be created as soon as the edge of the cube is crossed while pointing at it. This 'drag and copy' procedure gives the user a very sketchy feeling which is best indicated as "painting boxes in the sky".

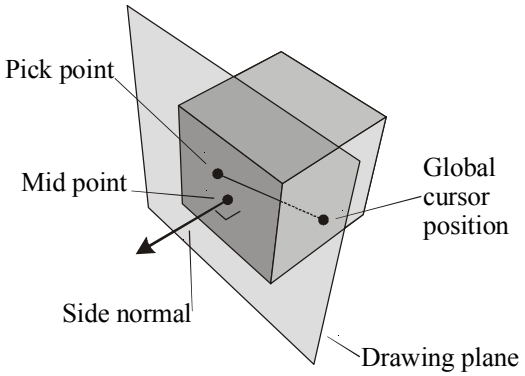

Figure 3: *Drag and Copy*

To determine if an edge of a cube is crossed while moving the cursor is a straightforward calculation knowing which face of the cube is picked. The vector pointing from the Pick point to the Global cursor position is called the Drag vector.

3.5 Edge Drag

The problem of a low-level data model, such as used here, is that operations on a higher level require a tedious effort. One example is dragging edges consisting of cubes. We have found a simple though very effective operator that allows a user to interact at a higher level, namely Edge Drag. With Edge Drag you can move one cube while at the same time moving all cubes left and right or above and below, depending on the dragging direction.

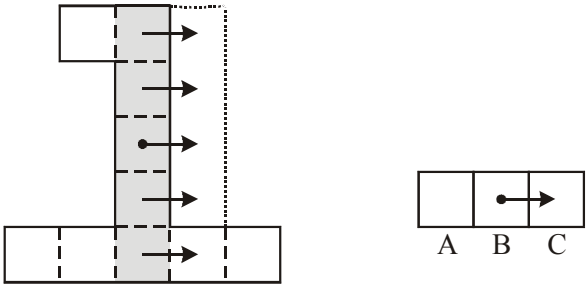

Figure 4: *2D Edge Drag*

The Drag vector together with the Side normal will determine the orientation of the cube (what is left, right, up, down, front or back). The Edge Drag algorithm is explained in 2D [Figure 4]. It consists of two steps, namely

Cube selection step:

First all cubes that will be moved are selected. In the left part of Figure 4, these are all connected cubes above and below the cube that is being dragged.

Delete-move and copy step:

For each cube that is selected in the first step do the following: The cell in front of the cube (see the right part of Figure 4) is called C and the cell behind it is called A.

1. If cell C contains a cube, remove it.
2. Move the cube in cell B to cell C.
3. If cell A contains a cube, make a copy of it in cell B.

As a result, the edge is dragged while maintaining connectivity with the other edges. An applet that demonstrates the concept can be found at the web [www.win.tue.nl/~vanwijk/dragedge/].

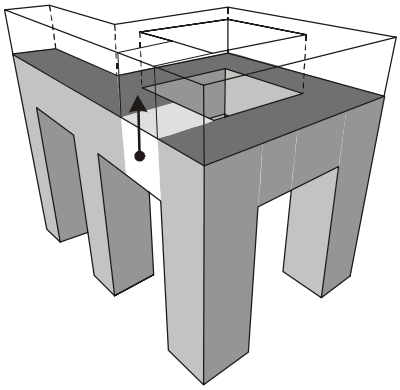

Figure 5: *Edge Drag*

The Edge Drag used in DDDoolz is a 3D version of the Edge Drag explained above [Figure 5]. From the starting cube, all connected cubes that are in the plane defined by the Drag vector are selected. After that, the delete-move and copy step will be executed as in the 2D version.

Note that the user never has to specify which (part of) the shape will be modified, the system detects this automatically.

4. System

The functionality of DDDoolz is arranged in four modes, namely: Sketch, Modify, Navigate and Orientate. Other functions include Move, Delete and Color. In the next paragraph the most important functions of DDDoolz are explained.

4.1 DDDoolz Functions

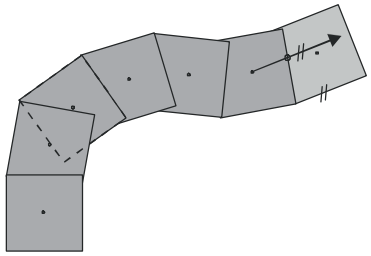

Figure 6: *Curve mode*

Sketch mode, using the Drag and Copy algorithm, is decomposed into two sub modes, namely Straight and Curve. In Straight mode a new cube is created next to the previous one. The copy direction is determined by the sketch device drawing direction over the cube side. In Curve mode the Drag vector will determine the new cube's orientation. The new cube shares the outer edge with the previous one, and it is aligned with the Drag vector.

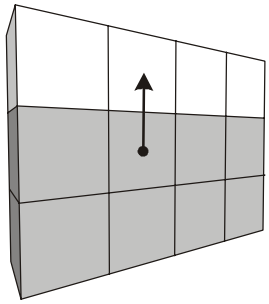

Figure 7: *Modify*

Modify will invoke the Edge Drag algorithm as soon as the cursor enters free space and if at least two rows of cubes are present perpendicular to the drawing direction. Instead of increasing a structure, the number of cubes can also be decreased by dragging the cursor in the opposite direction over the existing cubes. If only one row of cubes is present, then Modify will act as a Move operation on the cubes.

Navigate mode is decomposed into a Walk mode and a Fly mode. In Walk mode the viewpoint is set to a fixed distance to the ground whereas in Fly mode it is not.

In Orientate mode an individual cube can be rotated around the selected Side's midpoint. From there new strokes can be made in this newly set direction. Move mode allows for moving a cube continuously or over a snap distance equal to the cube's edge length. In Delete mode cubes that are pointed at will be removed from the scene. To prevent undesired deletion of obscured cubes, only those cubes are deleted that are in the same plane as the initially picked cube.

The Color menu offers a set of color buttons that will instantly set the color of the selected cube to the new color. Re-coloring is achieved in Sketch mode by moving the cursor of the cubes starting form the cube with the desired color.

4.2 Implementation

DDDoolz was developed using WorldUp from Sense8. In WorldUp Input handlers are available for an extensive set of commercially available input devices. A 3D scene is constructed as a hierarchy of geometrical objects. WorldUp has a few geometrical object types of its own (Block, Cone, Cylinder, Sphere) and it can import geometries in 3DStudio format and other formats. Special objects such as Viewpoint and Window are available for displaying the scene. To each object a script can be attached that will perform a specific task. Tasks will be executed in every rendering cycle of the scene following the objects hierarchy.

WordlUp comes with a player that can be distributed freely. The latest version of DDDoolz is almost completely implemented with WorldUp functionality including buttons for switching Mode. Consequently DDDoolz can be distributed as shareware and even be adapted to different hardware platforms with minor changes.

4.3 Data Exchange

DDDoolz models can be stored in its own format and reloaded. DDDoolz can also import externally created objects in 3DS format. Imported objects are treated as Imported Geometries in DDDoolz. An Imported Geometry cannot be edited, it only can be moved or deleted. An import script (VBA) has been developed to import DDDoolz models into Autocad. This facility allows for post editing with Autocad and 3DStudio. For instance the Autocad unify operation can be used to group all cubes into one single Solid.

Communication with other packages is not our primary research goal but is supported through the use of the design information database implemented as a feature model [9].

4.4 User Interface

Figure 8: *User Interface*

Since paper and pencil seem to be unbeatable, we considered using the familiar stylus interface in a new setting, namely sketching cubes in space instead of strokes on a piece of paper. For this purpose a touch sensitive Wacom LCD sketchpad is used. Making strokes on the LCD screen results in creation or modification of the cubes that constitute the design. Alternatively a 2D mouse can be used.

Use of the non-dominant hand for holding the object is a very natural way of physical object modeling. Also in the case of virtual object modeling this method has demonstrated quite good results [5]. For this purpose a Logitech trackball is used. Rotating the trackball results in rotating the DDDoolz model around the midpoint of the bounding box of the model. Zooming in and out is supported by pressing the trackball buttons. Alternatively the keyboard keys can be used. Rotating the model and sketching can be performed concurrently, which gives a good control over the design system.

Command input is implemented by 10 buttons in de lower border of the application window. In addition, voice recognition can be used for changing color.

5. DISCUSSION

5.1 Limitations

DDDoolz has been developed over a period of approx. one year. Since the first responses on the prototype system were very positive it was decided to make it more robust and to put it into practice. Comparing the design system objectives listed in section 3.1 with the final system, the following shortcomings can be noticed:

The interface is not fully command free. Since the system is a research prototype it was decided first to focus on the design metaphor and design functionality and thus to postpone the research for a command-less interface. However, division of viewing control and modeling operations over the left and right hand already significantly decreases the number of mode switches.

The shapes that can be created have the cube as its elementary object. Subtle curved planes or deformations of surfaces are not possible. This characteristic though is rather an advantage then a drawback because the granularity of the kind of shapes that are created in the early design stage is usually quite high.

The final model consists of a large set of individual cubes that don't have any coherence. In fact though, the complete architectural model most often can be subdivided into building blocks. Therefore a smooth transition from the individual cubes to Building Elements is required. As a designer you wish to move back and forth between both representations (Cubes and Building Elements) without being aware of the geometrical representation.

In this paper the connection with the design database is not discussed but is nevertheless implemented [9].

5.2 Experiences

DDDoolz has been used in a first years CAAD course at the Eindhoven University of Technology [www.ds.arch.tue.nl/Education/courses/7m063/results/2000_1]. As we had hoped for, the students had no problems using DDDoolz despite the very limited time. Some students took the sketch functionality quite literal creating quite messy, though sketchy shapes, with little architectural meaning. For a global sketch of a building mass the optimal cube dimension varies between 0.3 to 1.0 meter. Smaller cubes require too much effort to create the model and will decrease the system performance. Bigger cubes don't allow any shaping of the model. As soon as dimensional accuracy is pursued, DDDoolz will fall short. As a consequence DDDoolz designs will give the impression of a sculpture rather than an architectural design [Figure 9]. This characteristic however prevents the student from being distracted from the mass design by structural considerations.

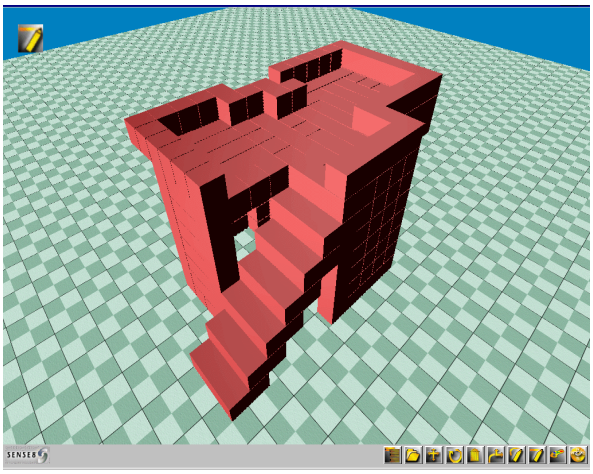

Figure 9: *DDDoolz design example 1*

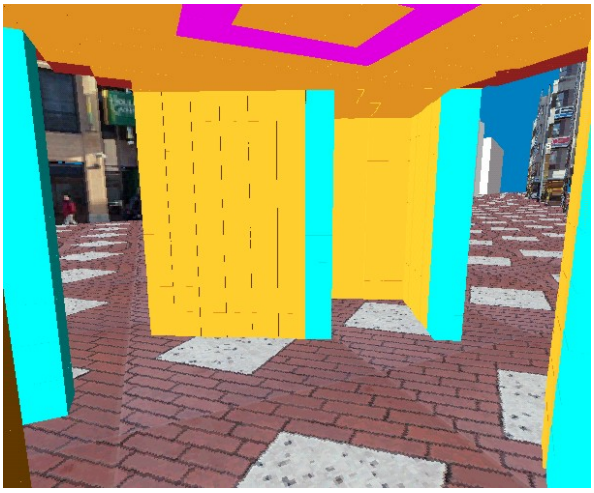

Figure 10: *DDDoolz design example 2*

In Figure 10 the individual cubes that can be moved, deleted or extended are quite well visible. The design task here was to create a mass model of a simple building in a given, existing environment. Fitness of the model in the environment can be verified immediately by the student and the teacher.

DDDoolz has also been tested by a large architectural office in the Netherlands on our request to explore application in practice. Here, the architect used DDDoolz for designing masses corresponding with rooms. The architect liked the very intuitive interface and the ease of viewing the design. Major restriction that was reported was the impossibility of changing the cube's size. Working on a spatial plan [Figure 11], he wanted to adjust the cube (i.e. the room) dimensions to the areas that follow from the client's brief.

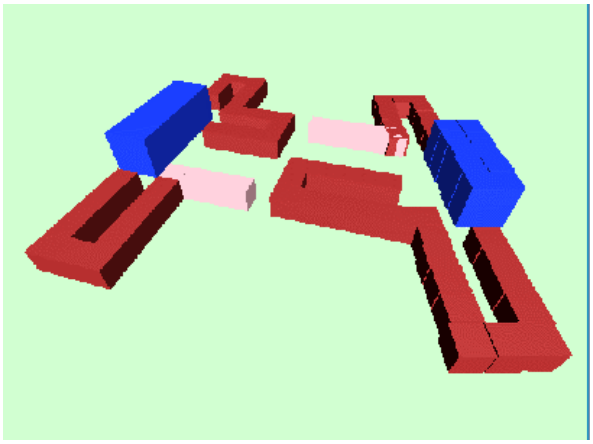

Figure 11: *Cubes used to represent rooms*

Interesting in this case is that DDDoolz was not meant to be used in this way. Cubes are considered as the building block for creation of masses. The architect however, related cubes to rooms.

5.3 Conclusions

Comparing the design results with 3D models created with CAD packages it is obvious that DDDoolz offers a quite new approach to modeling in the inception phase of architectural design. The strength of DDDoolz is in its simplicity and in the abstractness of the design.

5.4 Planning

Based upon the experiences the following system extensions can be listed,

Functionality:

- Groups

  Select cubes into groups. A Group can be inferred into a Building Element (e.g. a door) and have its own (non-cubical) geometrical representation. Having groups also allows for execution of the Edge Drag function for a group, thus offering more refinement.

- Change cube dimensions and properties

  Change height, width, and length by dragging a cube's side. Adding textures and setting transparency should be manageable for each Cube or Group.

User Interface:

- Command input by pen

  Take advantage of the pen pressure as in input parameter. Pen pressure can be used for instance to detect Modify (hard) and Sketch (soft).

  Different pens can be used, each with its own physical outlook corresponding with a specific operation (i.e. command).

- Desk-CAVE

  Using low budget LCD projectors a CAVE can be constructed around the desktop. The Desk-CAVE projection screens and the desktop can be used for projecting the designed model in its urban environment. Secondly the screens can be used for displaying different views on the design, such as structural engineering, building physics, etc. Thirdly the desktop projection can be used as a command interface in itself by tracking pen or hand movements [2].

6. REFERENCES

1. H.H. Achten, and B. de Vries. DDDoolz: A Virtual Reality Sketch Tool for Early Design. *Proceedings of The Fifth Conference on Computer Aided Architectural Design Research in Asia*,451-460, 2000.

2. D. Aliakseyeu, J.B. Martens, S. Subramanian, and M. Vroubel. Visual Interaction Platform. *Proceedings of INTERACT2001*, (forthcoming).

3. B. Buxton. An Introduction to the Special Issue on Interaction in 3D Graphics. *Computer Graphics: The SIIGRAPH Quarterly*, **32**(4), pp 43-44, 1998.

4. D. Donath, and H. Regenbrecht. Using immersive Virtual Reality systems for spatial design in

architecture. J. Verbeke et al. (eds*), Proceedings of the 2nd AVOCAAD Conference*, 1999.

5. M.W. Gribnau. Two-handed interaction in computer supported 3D conceptual modeling. *Ph.D. Thesis, Department of Industrial Design*, Delft University of Technology, 1999.
6. M.D. Gross, and E. Yi-Luen. Ambiguous intentions: a paper-like interface for creative design. *Proceedings of UIST '96*, 183-192, 1996.
7. T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A Sketching Interface for 3D Freeform Design. *Proceedings of SIGGRAPH '99*, 409-416, 1999.
8. D. Kurmann. Sculptor: How to design space? T. Sasada et al. (eds.), *Proceedings of The Third Conference on Computer Aided Architectural Design Research in Asia*, 317-325, 1999.
9. J.P. van Leeuwen. Modeling Architectural Design Information by Features: An approach to dynamic product modeling for application in architectural design. *Ph.D. Thesis, Department of Architecture, Building and Planning*, Eindhoven University of Technology, 1999.
10. R.C. Zeleznik, K.P. Herndon, and J.F. Hughes. Sketch: An Interface for Sketching 3D Scenes. *Proceedings of SIGGRAPH '96*, **30**(4), 163-170, 1996.