Blender, an Open Source Design Tool: Advances and Integration in the Architectural Production Pipeline

Theodoros Dounas¹, Alexandros Sigalas²
¹Aristoteleio University of Thessaloniki, ²University of Thessaly
¹dounas@gmail.com, ²alxarch@gmail.com

Abstract: We examine an open source 3d suite of tools called blender, as a tool for architectural design. The unique features of blender are examined in terms of ease of use and integrated nature since blender incorporates a simulation engine and a game engine that can be used creatively in the design process. The unique data structure of Blender is examined with the features and work flow that this structure brings in the design process. Also a simple comparison is made between Blender and 3ds max in terms of features and workflow as visualization tools together with an assessment of a two year seminar that took place in the Department of Architecture, in Volos Greece.

Keywords: Integrated design; open source CAD.

Blender, an open source 3D tool

Advances in CAD in the last 20 years have shifted the design paradigm from synthesis to integrated design where the design process is supported and informed from various disciplines simultaneously. This paradigm shift has been expressed in the recent efforts in developing supposedly highly integrated BIM tools, where a single geometric model holds all information about the Building. This shift from drawing to modeling tools has created a chicken-and egg problem in the design process, where the tools the architect uses supposedly inform the design process without the design process actually being integrated in nature. The tool we present in this paper provides a single tool for the integrated design approach: Blender is an open source software suite used in modeling, animation, rendering and compositing. Its use as a design instrument relies in its generic tool set, where design complexity is built from combinations of simple functions and data containers.

Its open source nature development model ensures that the user has always access to the source code used to produce Blender and also enables the user to customize Blender extensively or add new previously nonexistent functionality.

Analysis of Blender

There are 3 fundamental principles in blender: The object – data block information model; the modal and context-aware nature of the UI; and the modular behavior of its tools.

The object data handling is the basis of data storage inside file and data organizing in the workflow. The idea is that all information stored in a file and edited by the user is encapsulated within an object container. Duplication of information (or multiple usages) happens by linking a single Data block to multiple objects that share the same data but are independent instances each with its own levels of autonomy. The simplest example would be
the default cube. An object container called “Cube”, is linked to a Data block of type Mesh which holds geometry data for the object, and that Data block is linked to another Data block of type Material from which it gets shading parameters which in turn links to a Texture Data block which links to an Image Data block, etc. (Figure 1)

This model of data arrangement allows for great flexibility in data exchange between files and within a single file but more importantly, acts as the groundwork on which the modality of the UI is built upon. It also allows for data on one object to be used as control parameters for manipulation of other objects via hierarchical structures or other composed systems of interaction between them.

Blender’s UI is modal according to the paradigm of Jef Raskin - the humane interface. This modality is expressed in the interface through the various window types, each of which has a specific role in the architectural production workflow, but also reacts to reflect relative data to the current action the user is performing. Selection and arrangement of the displayed window types provides modular workspaces each customized to serve the task at hand. One of the ergonomic decisions behind the GUI is to enforce non-overlapping windows, a decision which many 2D and 3D graphics packages have reverted to lately. A function enhancing the UI is that all commands inside the software are common for all window types. The G-R-S keystrokes for example relate to Grab, Rotate, Scale commands that are active in all Blender window types and not just the 3d view window. That way the user that has the 3d view and IPO animation curves windows open can use the same keystrokes along with the same constraints in both windows. This fact helps the user to scale the learning curve of blender and allows for enhanced speed in its work flow after the user has learned same basic commands and keystrokes.

Although Blender tries to include all functionality needed for the production of 3D content ranging from still images to full movie productions and
games, it follows the UNIX philosophy internally by providing simple tools that are easily combined into systems that act as specialized tools. This open approach allows to solve design problems through analysis of the problem to simple rules and synthesis of form into a specific synthetic tool that provides the desired level of control and is editable at all levels. These can be further combined to create complex systems of interacting information that would have been impossible or impractical to have as pre-defined functions.

**Object – edit mode**

The most prominent example of the modal nature of Blender UI is the dual mode handling of geometry objects. They can be handled on two levels: object level and geometry level. On object level, properties of an object such as position, scale, rotation, hierarchy can be manipulated and on geometry level direct access is given to the form of objects by editing of the mesh data. The majority of the design process in Blender happens within a constant switch between these two levels by switching from the so-called Object Mode to Edit Mode and back. This method of design enables a clear separation of abstract ideas behind a design and the actual design data something rarely achieved in design software. This helps the designer organize her thoughts “on the fly” as she designs and also acts as the top-down – bottom – up ladder of the design process.

Following a bottom-up approach we will analyze the edit mode first. Edit mode is the level on which all direct geometry data manipulation happens in Blender. So far Blender focuses on a polygon modeling / subsurface toolset with other modeling options available (curves, surfaces, blob mesh) but not as featureful as polygon modeling ones.

From a typical poly-modeler’s point of view, Blender has great flexibility with many selection modes, proportional editing, special tools for vertices/edges/faces, auto-fill functions and more. From the designer’s perspective, blender edit methods use simple design helpers that can be combined in various ways to provide the desired parameters for a specific manipulation. Such helpers include the 3d Cursor (a reference point in 3D Space), snapping modes, on-demand axis constraints, numeric input, custom coordinate reference systems, semantic selection behavior, and interactive feedback for dimensions/angles/areas on transformation, transform manipulators (gizmos) and free hand transformation.

Following the principle of modality, design helpers come into play based on context of user actions thus eliminating most of the need for “planning ahead” before making a transformation. Following the principle of complexity through combination of simplicities, many usual restrictions are not explicitly defined but produced by making use of multiple helpers in parallel. (Figure 2)

During the transformation, these helpers can be changed, recombined, or reset until confirmation, providing a fluent non-linear workflow. As Blender developers focus on artistic production though, it lacks some of the advanced snapping features commonly found on CAD software and workarounds are still needed to implement some common CAD commands.

**Building custom tools**

**Animation and rigging**

The Designer can use Animation tools: key frames, paths, IPO curves, dupliframes -dupliverts, modifiers in a structured way inside blender expressing relationships between objects. These relationships constitute parameters that inform about the design process. Both geometric data and non-geometric data can be controllers or actuators of these relationships. For example a Parent and child relationship inside blender can inform about the span, curvature and individual elements of a bridge. (Figure 3)

In Pennington Roan Bridge case study a curve is used as path for an array/or dupliframe of individual bridge elements, as curvature for the deck of the bridge, and as length for the span of the bridge. With
a proper setup the designer can change the length and curvature of a single curve and the bridge can be made longer, bended in 3 dimensions, and also under certain conditions simulated in terms of load bearing and stress.

**Modifiers**

Blender has an extensive list of modifiers that transform geometry and animation. Although not all of the modifiers are effective during design procedures, the ones that have these characteristic in common are able to transform geometry of an object using another object’s data as input parameters, effectively making the other object a geometry controller. Even though this can appear to be a simple procedure, it has the added value of accomplishing complexity using simple modifications and combinations of simple parameters. That way the designer does not have to handle complex geometry directly but produce it through a series of adjustable operations. Modifiers in blender use the notion of derived mesh; each modifier accepts the mesh that derived from the previous operation to produce the final topology. The base geometry however is always editable through the usual Edit mode with realtime feedback on the results of the edits. This way the designer can manipulate the result on the base topology – geometry level, at the higher level of a modifier’s parameters or through geometry editing and transformation of a controller object. This creative loop allows for many design explorations since the designer does not need to “freeze” a model by applying a modifier to proceed and can edit the base mesh at any point without the modifier stack failing.

**Python API**

Scripting capabilities are provided through Blender’s Python API. The API can affect and control almost
all aspects of the software. It can add data/import export to and from other file formats, directly draw on the opengl viewport, add interactivity to objects, handle management of the Data blocks, and create animation effects and custom constraints. Python itself is easy to learn and fast to develop in. It offers a vast range of high quality libraries (scipy, numpy, PIL) and has bindings to the majority of open source libraries available to C. Complexity of python scripts varies from a simple script link adding interactive ruler measuring capabilities, to a material costs calculator with spreadsheet outputs, to a vector rendering solution.

Visualization capabilities
In the following table we compare architectural visualization characteristics of 3ds max and blender. The table has been compiled taking into account wide industry reviews, questionnaires and comparison studies in the architecture visualization industry and also the author's personal experience with both software. The table compares future to future characteristics of two software. It should not be used as a general way of deciding which of the two software is best in the field, as for specific needs each field in the table would have a different weight. (Table 1)

<table>
<thead>
<tr>
<th></th>
<th>Blender</th>
<th>3ds max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry editing</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Rendering speed(no GI)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Rendering speed (GI)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Animation (design)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Animation (presets)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Parametric design</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Parametric objects</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Modeling Speed</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Data exchange</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Plugins – extendibility</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Compositing features</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Interface use and speed</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Memory and CPU footprint</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Case studies
Shops and office building in Volos, Greece – Sigalas Alexandros, Belos Ioannis (10th semester project)
This project was one of the most demanding in the course of our studies as it enforced building regulations in addition to concept development and demanded high level of technical details in the final plans. It demanded constant adjustments during the course of the semester as new regulations were introduced to us per lecture. In addition to the normal regulations the site included a listed building. Blender constraints mechanisms and some scripting
provided the tools to develop the concept with constant feedback on the shape and coverage regulations and the building program demands. After concept development, interoperability with AutoCAD (used for 2D full detail plans) was constant to visualize and decide on changes introduced to the plans as the course progressed. Precision modelling helpers allowed to keep the 3D model “aligned” to the imported 2D plans and Blender scene management techniques such as multiple groups per object were used to ease navigation through the model for decision making. Compositing features were used for final visualizations and to keep render times low even with large image sizes. (Figure 4)

Kapnergati square urban shelters – architects: Dounas Theodoros-Soulakis Nikoalos // array-curve modifier - dupliframes on a path

The first case study presented is the third prize in an architectural competition in the city of Kavala, Greece. The competition was the redesign of the “Kapnergati” urban square in the city of Kavala. The Urban Shelters in the kapnergati square redesign competition are the only 3 dimensional objects designed for the competition. During concept development the requirement for the shelter to appear as “dancing” or “moving” emerged, a contrast to the rest of the urban square which was designed in a classic static fashion. Two parallel strategies were used to accomplish the effect of “moving” shelters: An array of a square shelter modified by a Curve modifier and a square shelter animated along a path with Dupliframes added. These two strategies were, but the array with curve modifier strategy was more efficient in terms of design control. In the end the animated path was used as a driver for the concept the designers were trying to implement and the array with curve modifier was used as the actual implementation of the design. This dual strategy is useful from a design standpoint since the designer is not restricted in a single wizard or parametric object

Figure 4
Sara & Mara Shops and Office Building in Volos, Greece

Figure 5
Shelters in Kapnergati square
to accomplish something but can develop his own “parametric” designs according to the desired result. (Figure 5)

Lex, Walsall - Urban Splash competition – architects Agrafiotis Nikolaos, Dounas Theodoros, Karagoriou Vasilios

The case study of the urban splash competition in Lex, Walsall UK was really simple as a concept: a single parallelepiped is animated from the road towards the water canal. Key frames were used for the animation while the parallelepiped moved from the “urban” side of the site to the “water” one, also moving in height during this transformation creating the floors. The external mesh for the building was created by rotating a single object controlling an array of vertical beams. (Figure 6)

Education

We taught blender as a general design tool during the spring semester the last two years in the Department of Architecture, Polytechnic School, in the University of Thessaly. The curriculum was established from a design perspective, incorporating theory and practice during classes. The class covered modeling, texturing and animation techniques and focused on topics such as correct topology, hierarchy and animation metaphors, materials – light interaction and the way these topics are handled in Blender. We chose to expose the students to some basic theoretical background of 3d computer representation as we felt that this is the best way to have the students being able to learn more on their own after the end of the course. This decision was also based on the rationale that the enforcement of proper modeling makes comprehension of the structural values of the designed object easier. Modeling and other workflows in Blender don’t try to hide the way the computer handles the data in 3D computer graphics. The designer is exposed to the mechanics of computer graphics, the knowledge of which is applicable in every 3D program. The modular approach of the modeling tools, as opposed to a single-button-specific-task one, helped the students to develop “design thinking” and use Blender as a decision-making tool for concept design and not only as a visualization tool. This head-on approach might be considered a weakness as the learning curve at the beginning has proven to be difficult. Students with no prior 3D
experience and ones who had knowledge of another 3D package but without any theoretical background had to change their mechanistic view of “which button does which task” and spent time to understand things like model topology and material shading behavior. This process pays out in the end but might dishearten the category of students who are “afraid” of computers in general.

Conclusions

Blender has still a long way to go in terms of percentage of adoption among architects, not because of lack of features or integration capabilities but because it is an open source software, downloaded free of charge and usually perceived as a mere toy. Its integrating features though along with the way it is structured allow for a wide range of uses inside the architectural studio, either in practice or in education. Although most users will be faced with a steep learning curve when using blender, scaling that curve will bring many benefits in terms of design thinking and concept development in the architectural work flow of the user. In the end, in our experience in mastering a software like blender the user becomes more comfortable with design tool making, whether that is a simple geometric model, an animated structure or a script enabled generative system.

References

Terzidis, Kostas; Jungclaus, Jan; 2007 - Predicting the Future: Open Source CAAD?, Predicting the Future