Architectural Design in Open-Source Software

Developing MeasureIt-ARCH, an Open Source tool to create Dimensioned and Annotated Architectural drawings within the Blender 3D creation suite.

Kevan Cress¹, Philip Beesley²
¹,²University Of Waterloo School of Architecture
¹kevan.cress@gmail.com ²philip.beesley@uwaterloo.ca

MeasureIt-ARCH is a GNU GPL licensed, dimension, annotation, and drawing tool for use in the open source software Blender. By providing free and open tools for the reading and editing of architectural drawings, MeasureIt-ARCH allows works of architecture to be shared, read, and modified by anyone. The digitization of architectural practice over the last 3 decades has brought with it a new set of inter-disciplinary discourses for the profession. An attempt to utilise 'Open-Source' methodologies, co-opted from the world of software development, in order to make high quality design more affordable, participatory and responsible has emerged. The most prominent of these discussions are embodied in Carlo Raitti and Mathew Claudel's manifesto 'Open-Source Architecture' (Ratti 2015) and affordable housing initiatives like the Wikihouse project (Parvin 2016). MeasureIt-ARCH aims to be the first step towards creating a completely Open-Source design pipeline, by augmenting Blender to a level where it can be used produce small scale architectural works without the need for any proprietary software, serving as an exploratory critique on the user experience and implementations of industry standard dimensioning tools that exist on the market today.

Keywords: Blender, Open-Source, Computer Aided Design, OSArc

OPEN SOURCE AND OPEN SOURCE ARCHITECTURE
In our current technological era a new architectural discourse has sprung up in an attempt to breath new life into the ideals of Participatory Architecture championed by the likes of Christopher Alexander, Giancarlo DeCarlo and Cedric Price in the mid 1900's. Re-Branded as "Open-Source" (a term co-opted from software developers) the Open-Source Architecture (OSArc) movement defines itself as "an emerging paradigm [...]" drawing from references as diverse as open-source culture, avant-garde architectural theory, science fiction, language theory, and others, it describes an inclusive approach to spatial design the collaborative use of design software, and the transparent operation throughout the course of a building [...] life cycle". (Ratti 2015) While this description leaves room for a lot of interpretations, the
flavour of OSArc that this work seeks to promote is one that is able to be freely shared and accessible to as many people as possible.

To understand the lofty ideals of the OSArc movement it is useful to understand the origins of the Open-Source software movement whose terminology the OSArc movement has co-opted.

**Open Source Software**
The beginnings of the Open Source movement can largely be attributed to the work of Linus Torvalds and his work on the Linux operating system kernel, which he released for free in 1991. This would spark other developers to begin releasing their source code for free, most notably Netscape would release their web browser (now known as Mozilla Firefox) and its source code to the public in 1998 and coin the term “Open-Source”. Importantly this was accompanied by ‘Copyleft’ licences like GNU GPL that are a manipulation of traditional copyright law to ensure that any derivatives of open-source code must also be released as open-source under the same license, a protection that these works would not have if they were to be released into the public domain.

**OSArc**
Carlo Ratti and Mathew Claudel’s manifesto “Open Source architecture” began life as collaborative Wikipedia page in 2011 that would serve as the kernel for an article published in domus that June. (Ratti 2011) After the article’s publication, the text continued to grow into the book that was eventually published in 2015. Ratti and Claudel (and a host of many other collaborative authors) propose that the next evolution of participatory design is one that will build on the success of the open source software movement by creating an ‘open-source’ Architecture. An architecture that can be shared modified and redistributed but will ultimately be composed by what they refer to as the “Choral Architect” who weaves together these disparate ideas of the populous into a harmonious whole.(Ratti 2015) The text itself and its successful collaborative authorship would seem to be a positive indicator of the potential of this mode of working. However the translation from Open-Source text to ‘Open-Source’ Architecture is quite challenging.

**The Nature of ‘Sources’**
Open-Source Software relies on the fact that Software can be collaboratively shared modified and reproduced solely from its source code. In basic terms a software’s source code is plain text written in a human readable programming language that can be compiled to produce the computer readable machine code that is the software. It’s important to note here that, because a software’s source code is simple plain text, anyone with access to a computer and the internet can obtain, modify, and compile the code to a usable program without the need for any commercial tools. This means that the barrier to entry for an open source project is incredibly low. Unfortunately this is not the case with OSArc (Vardouli 2012).

Most OSArc projects treat the virtual representations of a building (its 3D models, BIM Data, CAD drawings etc.) as the ‘Source-Code’ of architecture, and utilize common Open-Source platforms like GitHub to share and manage the collaborative editing of this data. However, unlike software source code which is comprised of plain text, this Architectural data requires the use of specialized commercial software to read and edit. OSArc projects have excelled at the sharing of their Architectural data over the web, however the industry standard tools available for editing it are commercial, closed-source, and proprietary which imposes a serious barrier to entry. The cost of these industry standard tools poses a substantial problem for OSArc, as the success and longevity of Open Source Projects is typically directly correlated to the size of the community it can engage in development. For this reason, this work argues that for an OSArc project to be successful (and Open-Source, in the truest sense of the term) then the set of tools necessary to interact with the Architectural data of that project must be open-source as well.
**WikiHouse**

Wikihouse, an ‘Open-Source’ architectural project founded by Alastair Parvin and Nick Lerodiaconou in 2011 is a contemporary attempt to leverage the design talents of the masses to produce an innovative system for affordable housing. Their unique structural system WREN v4.3 consists of an assembly of pieces that can be cut from standard plywood sheets on a CNC Machine. All of the plans are released for free on the organization’s GitHub and are licensed MPL2.0 meaning collaborators are free to download and modify them as they see fit.

Wikihouse has seen a good deal of media coverage, with several built projects, and general praise for their advancement local digital manufacturing and low-income housing and construction. However, The WREN structural system as it exists on GitHub, throughout its 58 revisions, has only 2 contributors both members of the Wikihouse foundation. The smaller Microhouse system fares slightly better, with 3 contributors. The ‘source code’ that they’ve shared for the WREN system is based in Rhino’s Grasshopper, this means a minimum buy in of $995 for an individual to engage with the project, and McNeils Rhino is on the cheaper end when it comes to architectural software. To an established architecture firm $995 may seem like a nearly insignificant cost, but it’s important to remember that the lifeblood of an Open-Source project is individual, volunteer participation, which means that for a project to grow and flourish the barrier to entry must be as low as possible, and this is a difficult problem to overcome when looking at the landscape of architectural software available today.

**BLENDER AND MEASUREIT-ARCH**

Our tool, MeasurIT-ARCH aims to provide an Open-Source alternative that OSArc projects can utilize to ensure that their barrier to entry is as low as possible. It is a dimensioning and drawing plug in, based on the MeasureIT tool created by Antonio Vázquez, developed for Blender, the Open-Source 3D creation suite started by Ton Roosendaal in 1995.

**Blender**

Blender started life as a proprietary tool for NeoGeo a Dutch animation studio co-founded by Roosendaal. From 1998 - 2002 Blender continued to be developed by Roosendaal under the company Not a Number. In 2002, following Not a Numbers shutdown, the community surrounding blender, initiated by Roosendaal, raised 100,000 Euros to purchase Blenders licensing from the Not a Number investors and re-release it as an Open-Source software under the GNU GPL license [@foundation]. Since 2002 Blender has seen stable development and a robust community. While Blender has been developed with a predominant focus on animation, there are several aspects of Blenders design that make it uniquely suited to be used as an architectural design tool:

1. **Realtime PBR (Physically Based Rendering).** Blenders latest release features the EEVEE rendering engine. Which can simulate photorealistic, lighting and material characteristic (including reflection and refraction), while you work. This is a similar feature set to Lumion, a rendering tool frequently used in conjunction with SketchUp or Revit.

2. **Procedural Modeling,** more commonly referred to as parametric modeling in the Architectural industry, generates forms based on a series of rules and input parameters. In Blender this is accomplished through the modifiers system. Which takes a simple input mesh parameters and preforms operations to it to generate or deform it into something new. Blenders driver system allows the user to derive input parameters from virtually any property of any object in the scene. Jacques Lucke is currently in the process of developing this modifier system into a node based interface, similar to Rhinos Grasshopper, and node based Parametric modeling is already possible in Blender through the use of either the animation-nodes addon (also created by Lucke) and the Sverchok addon.

3. **Object MetaData.** While it is beyond the
scope of this work to add BIM functionality to Blender, it is important to note that Blenders data structures and code infrastructure are already well situated to handle adaptations to manage the kinds of object metadata required to facilitate a BIM workflow. Users can already define custom properties that are attached to an object and use these custom properties in conjunction with the Driver system to make them influence geometry. Objects can also be linked across blender files, meaning that a user can create collections of objects with customizable parameters that can then be linked into the current working file, in a workflow similar to Revit’s Families, or SketchUp’s Components. The user interface to perform these tasks is not particularly intuitive at the moment, however the basic functionality to carry out BIM-like tasks exists and could be improved in the future.

**Missing Dimensions**

That being said, Blender lacks any ability to produce working drawings. This is a major detriment to its possible adoption by the architectural industry or use in OSArc projects. Measurelt, an addon produced by Antonio Vazquez, added basic dimensioning tools to blender, but the addon has no support for Dimension Styles, or Line drawing, both features necessary to produce architectural drawings in an efficient manner. Measurelt-ARCH, the tool developed in this work, builds on Vazquez’s addon to add in this missing functionality, and optimize the user interface and tools for use cases typical to architectural practice.

**MEASUREIT-ARCH, FEATURES AND DESIGN PRINCIPLES**

Creating an Open-Source set of dimensioning annotation and linework tools means writing them from the ground up. This provides an opportunity to rethink the paradigms and systems that define these tools and how they are implemented in industry standard software packages, and try and do better. Each of Measurelt-ARCH tools, while inherently pragmatic in nature, try to take a critical stance in their implementation, subverting in subtle ways the expected modes of operation of dimensioning, annotation, and linework tools. At the core of each of these alterations is the driving principle behind Measurelt-ARCHs design, avoid enforcing reductions to 2 Dimensional planes, embrace and engage the inherent complexity and opportunity of 3 Dimensional space, and make use of its depth and context whenever possible.

**A Hybrid System**

For Measurelt-ARCH to be a real time, 3D dimension system it needed to fully integrate with Blenders 3D viewport systems. Measurelt-ARCH is able to do this by utilizing Blenders robust Application Programming Interface (API). Application Programming Interfaces are a relatively common feature of large software applications, they provide users the ability to access certain aspects of the software through a programming language. Revit, Rhino, and SketchUp all feature API, however their depth and usefulness vary. Typically the scope of an API allows users to write addons for the automation of repetitive tasks,
Figure 2
MeasureIt-ARCH
Draw Code Diagram
and the importing and exporting various filetypes. Chaining together the existing toolset of a software can be substantially powerful, Rhino’s Grasshopper plug-in is a prime example of this, however Blender’s API takes things a step further, providing low level access to some of the software’s core functionality. This is essential for MeasureIt-ARCH as it allows the addon to create brand new tools and elements that integrate directly with the parts of the software that draw the 3D environment to the screen.

![Blenders drawing system can utilize 3 distinct render Engines. Eevee, A realtime engine comparable to the rendering capabilities of software like Lumion, or Unreal-Engine, Cycles, a raytracing renderer comparable to renderers like v-ray or maxwell, and the Workbench engine, a simple and fast engine to draw the working viewport. While the Workbench engine is the predominant viewport render engine, any of the 3 engines can be used in real time while working on a model.

Overtop of the main render engine (either Eevee, Cycles, or Workbench, depending on which the user currently has enabled). Blenders Overlay system draws other elements necessary for interacting with the 3D environment. MeasureIt-ARCH functions in the same way as the Overlay system. This means MeasureIt-ARCH elements can be layered over any of Blenders render engines for a variety of visual styles. Even through MeasureIt-ARCH is an Overlay, in the sense of the render pipeline, it still accesses the same 3D information as the main Engine allowing its elements have the appropriate depth ordering.

This Hybrid system allows MeasureIt-ARCH to take full advantage of Blenders rendering capabilities, and can blur the distinction between rendered image, and dimensioned drawing, if desired.

**Automated Placement of dimensions**

Current Industry tools like Autodesk’s Revit and McNeels Rhino deal with dimensions only in the 2D plane. The creation of Dimensions in these packages requires the user first to specify a “Work Plane” (or CPlane in Rhino) onto which dimensions will be projected. This approach one key advantage, as the number, and complexity of computations required to properly locate and display the dimension element in 2D are significantly reduced. Dimensions projected on a 2D work plane then can be quite efficient when the desired end product is a 2D plan, section, or elevation drawing of predominantly orthogonal geometry. However 2D work-plane dependant dimensions begin to cause issues in efficiency when working with 3D isometric or axonometric views, as the work-plane needs to be manually redefined by the user to add dimensions to each axis. Things become especially...
problematic when dealing with non-orthogonal geometries any complex geometric form requires multiple unique work-planes to be thoroughly dimensioned, and any modifications that move the geometry off of the dimensions work-plane will result in the models connection to the dimension to break and be deleted (as is the case with Revit), or become useless nonsense (as is the case with SketchUp). 2D dimensioning workflows present other limitations as well, such as Revits inability to display dimension elements in perspective views. This layer of abstraction for the sake of computational simplicity impacts the usability and efficiency of these dimensioning tools. Can we remove the work plane and it’s limitations, what new possibilities does this allow, and what new challenges does it present.

Firstly lets examine how the computations required to a dimension change when moving from 2D to 3D. Adding a dimension in 2D is fairly trivial. We need to draw some extension lines perpendicular to the points we’re measuring, and a dimension line with some text parallel to the points we’re measuring but offset by some amount. The bulk of the computational work here is involved in finding the correct perpendicular direction. For a 2D line A defined by values (x,y) its two perpendicular lines would be (y,-x) (a 90 degree rotation counter clockwise) or (-y,x) (a 90 degree rotation clockwise). With only two options to choose from a piece of software can very simply rely on sampling the users mouse movement to decide which of options to choose, and how long to make the perpendicular extension lines.

Moving up to 3D things become more challenging, while in 2D space there were only 2 possible perpendicular lines to choose from, in 3D space there are an infinite set of perpendicular lines. Trying to choose the users intended option from this infinite set using only mouse input would be unreliable and frustrating for the user. This leaves us with 2 options, maintain the status quo and reduce the problem back to 2D space by having the user pre-define a work-plane, or utilize more information from the 3D environment to select the correct perpendicular direction without any user intervention.

To select the correct perpendicular direction without any user intervention we first need to develop a series of rules defining how dimensions should ideally be placed:

1. Dimensions should never be placed inside of an object
2. Dimensions should remain perpendicular to one their adjacent faces.
3. Dimensions should be as close to parallel as possible to the desired viewpoint to ensure readability.

Implementing these rules requires information about the geometry surrounding the dimension and the desired view direction

Using this information, and some linear algebra it’s possible to produce an algorithm that properly places the dimensions without manual user input. The resulting algorithm, as it is currently implemented in Measurelt-ARCH can place Dimension elements in a useful and predictable manner on orthogonal and non-orthogonal geometries, and automatically adjust its location as the geometry is modified. Also, because the correct placement relies only on the 3D context and not user defined inputs, multiple dimensions can be added simultaneously through a single use of the dimension tool, a feature that, even if possible, has never been implemented in a work-plane based dimension system.

The downside to this method, as mentioned earlier, is that it significantly more computationally intensive. It requires querying nearby geometry, a series of matrix multiplications, as well as several conditional checks to ensure the algorithm functions as expected even in special cases when some of the expected information is not available to it. That being said modern computer systems have more then enough capacity to deal with this added complexity. To put this in perspective, adding 32 dimension elements with the dimension tool in Measurelt-ARCH takes just under 3 seconds, a little less that 0.1 second per dimension. While this is relatively slow oper-
ation in computational terms, it seems like a reason-
able trade-off when compared to the user input time
required to add each dimension individually in con-
ventional software packages.

**Realtime Dynamic Linework**
MeasureIt-ARCH lines are aware of their context in
the 3D scene, allowing them to automatically change
how they are displayed (based on the user defined
line style), if they are hidden by other geometry, or
if they represent the silhouette of an object. This
means that easily readable line drawings and dia-
grams can be produced without the need for the Ar-
chitect to manually define each individual lines prop-
erties.

**MetaData Based annotations**
MeasureIt-ARCH adds an annotation system the uti-
lizes Blenders custom object metadata. This means
that objects can be quickly given identifier tags, man-
ufacturer information, or any other useful metadata,
and this information can be used to populate draw-
ing annotations. A toolset to collect and output
schedules from this metadata is planned to be com-
pleted in the coming months, but is not yet imple-
mented.

**Style System**
Lines, Annotations, and Dimensions can all utilize
common Styles. Styles can be picked an assigned
during the elements creation. The style system, like
any CAD style system make it easy to maintain con-
sistency across drawings.

**SOFTWARE TESTING**
In addition to the development of MeasureIt-ARCH,
this work also involves the testing of Blenders abili-
ties in real world design scenarios. Ongoing work
for the Lodge at Pine Cove serves as a real world
testing ground not only for MeasureIt-ARCH but for
Blenders overall capacity to be utilized in architec-
tural projects. Working for a real client ensures that
the software holds up, not only in a theoretical sense,
but is usable even under time pressure and real-
istic deadlines and time budgets. Developing the
MeasureIt-ARCH in a cycle of software improvement
and design implementation helps ensure that the
changes made are practical in their intended use

case.

This testing has motivated improvements in line
quality and and the organization and implementa-
tion of the style system in order to improve the ef-
ficiency of the working process. It has also exposed
vulnerabilities in the text orientation system, which
will be the focus of development for the upcoming
months.

**COMMUNITY ENGAGEMENT**
For MeasureIt-ARCH to survive and grow it needs to
engage with both the Blender community, and the
Architectural community using methodologies that
are appropriate for each of the respective groups.
For an Open-Source project to survive, there is nothing more important than its community. Rather than solely relying on traditional academic modes of information collection and dissemination, this project aims to truly embrace a the Bazaar style of open source Methodology described by Eric S. Raymond (Raymond 1999), sharing the work through modern media outlets into the communities it seeks to engage with as often as possible, to generate direct feedback about the ideas proposed, and promote discussion. For engaging with the Blender community Measurelt-ARCH has been utilizing GitHub, to keep the management of the project as open and transparent as possible, and to provide a place where those interested in testing and providing feedback on the project can access it at any stage in its development. To share the project throughout the community and direct those interested to the GitHub page, tutorials on the use of the tool, and time lapses of it in action on demonstration projects are being recorded, published on YouTube and shared through blender community. The first round of community engagement, published on March 24th, has resulted in some reasonably substantial feedback which has lead to major bug fixes in the Measurelt-ARCH software, including resolving, compatibility issues with the MacOSx operating system, save and load issues breaking the text drawing system, and line drawing issues caused by differences in OpenGL support across graphics hardware.

In addition to bug fixes, community engagement has resulted in the creation of a public project planning board on the GitHub platform to manage and prioritize community feature suggestions, as well as my own development plans for the Measurelt-ARCH tool. The Measurelt-ARCH GitHub can be found at (https://github.com/kevancress/Measurelt-ARCH)

REFERENCES
Ratti, C 2011, ‘Open Source Architecture (OSArc),’ Domus
Ratti, C and Claudel, M 2015, Open Source Architecture, Thames \\& Hudson, New York, New York
Raymond, ES 1999, The Cathedral \\& the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O’Reilly, Beijing ; Cambridge, Mass
[1] https://www.blender.org/foundation/history/
[2] https://www.youtube.com/watch?v=QL_ArANpsVU
[3] https://www.youtube.com/watch?v=lHI78DB8bs