URBAN PINBOARD

Establishing a Bi-directional Workflow Between Web-based Platforms and Computational Tools

MADELEINE JOHANSON1, NAZMUL KHAN2, ROB ASHER3, ANDREW BUTLER4 and M. HANK HAEUSLER5
1,2,3,4 Cox Architecture Sydney, Australia
1,2,3,4 {madeleine.johanson|nazmul.khan|rob.asher}@cox.com.au
5 Computational Design, University of New South Wales, Australia
5 m.haeusler@unsw.edu.au

Abstract. Architecture is heading towards a future where data is collected, collated and presented in a dynamic platform. There is a potential for many standard processes in the industry to become automated, such as the site analysis process. Streamlining aspects of the design process allows architects to pay greater attention on creative design solutions for their buildings and less time engaging in complex, time consuming analytical programs. Urban Pinboard, a web-based GIS platform, promises to establish a bi-directional workflow between web data depositories and computational tools through the medium of a website. By doing so, the website allows users with minimal experience in computational processes to be engaged in the utilisation of these large datasets. Through the automation of these processes, relationships within the built environment industry can excel, leading towards performative driven designs.

Keywords. Urban Planning; Computational Urbanism; Data-driven Design; New Workflow Models; Software Development.

1. Introduction

Digitised data has been described as a dynamic and flexible technology that is fundamentally transforming our society as outlined in ‘The Second Machine Age’ (Brynjolfsson 2014). Reflecting upon industry challenges that architecture faces in the digital age, information data is scattered throughout the internet, estranged from correlation and relationship. This is a challenge for the AEC (Architecture Engineering Construction) profession as, due to its large number of different stakeholders, large quantities of data are involved in completing a project. Yet, if this
data was unified and composed, it could greatly assist architects in their work and support their productivity, freeing them from manual data collection and analysis and enabling them to focus on different aspects of the design process.

In the built environment, data is mostly stored in Geographic Information System (GIS) models, which store and display spatial information in relation to specific positions on the earth’s surface. Here, large data sets of urban and geographical information exist, however, a common issue for architects and designers is that these data sets are either inaccessible (no open source GIS data is yet available in Australia - however data sets are being increasingly built in the US), or are associated with large costs when embedded in commercially available GIS software packages. The background for these limitations is that data sets have only recently been made public through open source and through a new Australian government e-planning initiative. These large data sets of geographical information will become available for free, but on individual platforms, mainly each author of data will host their data on their website. example?

Consequently, one can argue for a need to collate this existing data within a single centralised data depositary (Picon 2010). This ‘need’ in the industry has been addressed in Australia through the emerging platform Urban Pinboard - a digital platform currently in development by the research team under commission and enabled through a ‘City Life Lab’ grant from the Urban Development Institute of Australia (UDIA). Urban Pinboard is a new tool for collaboration that connects the public, private and community sectors in a single conversation with innovative digital technology to drive smart city transformation. Urban Pinboard will be also unique and powerful because, unlike other one-way e-platforms in use around the world, it will allow these three sectors to not just access data but also exchange their ideas, thus creating a real and powerful feedback loop. Hence it can be considered as a library of open urban data, represented graphically in a global map interface. The platform creates an interface between unreadable datasets and its users such as government policies, survey information and development proposals. Consequently, we argue that Urban Pinboard has the potential to unlock communication barriers through technology, and produce accurate data flows where site potential can be determined through a dynamic platform. This notion is backed up by As and Schodek (2008) who state, ‘The way in which we visualize architecture, directly by graphical representation methods, affects the way we design and perceive architectural form and performance.’

2. Research Project

Yet, Urban Pinboard as well as other currently available open data sets on the internet are not linked to computational programs such as Rhino and Grasshopper. Thus, the multitude of data can often not be utilised via commonly used and readily available computational design tools, but by bringing the open data set platform and computational design tools together, one could unite and streamline current practices. Hence, the study aims to create a bi-directional pipe between web-based data depositories and the computational tool, Grasshopper. Having a website as the interface for computational processes would allow users, who aren’t necessarily trained in computational design or architecture, be able to control the
information that they want to see in a simplified, graphical format. An interface for decision making allows a process oriented experience for the user (As & Schodek 2008) and further as argued by Huang and Krawczyk (2007) ‘Ideas of what constitutes a design experience are expanding. Different opportunities for designers and consumers are opening up as control over design decision making is no longer entirely in the hands of the architect.’ Furthermore, creating an Application User Interface (API) for the pipe within Grasshopper would allow developers who are computationally skilled, to create customised scripts to apply to these large data sets. As a proof of concept, we will be creating a shadow analysis tool in Grasshopper, however the data ‘pipe’ is the central exercise (figure 1). From knowledge of Javascript and Python to construct the ‘pipe’ to Grasshopper, strings of data can be transported to one platform to another, used in Grasshopper’s analysis, and sent back to Urban Pinboard in a graphical format.

Figure 1. A diagrammatic representation of the bi-directional pipe, connecting web-based data depositories and Grasshopper.

3. Methodology

Drawing on knowledge in web-based rendering (Alttabtabai & Yan 2007), e-planning initiatives and the scripting languages JavaScript and Python (Kostadinov 2006), the research used an agile method to develop and test the proposed pipe in parallel and preparation of Urban Pinboard’s development. The objectives of the developed Python based pipe are to:

- Enable a user to select a specific geographical area with a mouse selection on a website,
- Feed data associated with the selected area via the ‘pipe’ to Grasshopper,
- Compute the result in Grasshopper,
- Send back the results through pipe to Urban Pinboard as rendered 3D objects.

As a proof of concept, the research is using the open-source environmental analysis plugin Ladybug for Grasshopper to create the shadow analysis tool. The paper concludes and argues that a direct link between these two platforms would benefit both, client and architect. This is based on the hypothesis that
Unlocks potential communication barriers established through technology,
- It produces accurate data flows where architectural sites potential can be determined through a dynamic platform, and
- Streamlines and analyses data sourced from online data depositories.

Hence, offering a new automated means of delivering urban design outcomes and providing more context about the geographical area selected by the user. This provides instantaneous, site-specific information that has the power to revolutionise the industry, streamlining previous practices.

4. Development Steps to Establish the Data Pipe

To develop the pipe, the application will use Python Flask (2016), a micro web framework to host the site. HTML and CSS files are linked in to create the visual display and functionality on the client’s side. Using Flask allows the application to use the flexible programming language Python to create the data pipe and transport information sourced from web data depositories.

4.1. STEP 1: ENABLE A USER TO SELECT A SPECIFIC GEOGRAPHICAL AREA WITH A MOUSE SELECTION ON A WEBSITE

At the time of the submission of the paper (October to December 2016) Urban Pinboard was not yet fully functional as the launch date of the beta version is scheduled for end of March 2017. Consequently the research team had to test its hypothesis on another, still similar big data depository platform. Based on research conducted during the development of UrbanPinboard the developer platform of Mapbox (2016) has been identified as he preferred big data depository source. Mapbox is used across the industry to visualize and create interactive maps. Using the Javascript library and the API of Mapbox, one can extract the information required to conduct the site analysis in Grasshopper. As our proof of concept, the research will be extracting the latitudinal and longitudinal of a clicked site to create a visual sunlight hour analysis.

Previously, only building footprints could be viewed in Mapbox, however recent development has enabled users to extrude these building footprints to represent their real-life counterparts. This building height data is yet only widely available in areas such as the US and the UK, so the research team has chosen to locate the case study project in San Francisco.

4.2. STEP 2: FEED DATA ASSOCIATED WITH THE SELECTED AREA VIA THE ‘PIPE’ TO GRASSHOPPER

When the user clicks a geographical point on the map, the website needs to save the Latitude and Longitude of that point. Using JavaScript, one can define a function to perform on a click action. Using the Application Programming Interface (API) of Mapbox one can then access the latitude and longitude at that point. Assigning this to a variable enables the user to use this data at a later point in the code. If the user decides to click elsewhere, the function runs again and the variable is assigned the new latitude and longitude.

As Mapbox delivers this as an object, one must reformat this data to a JSON
string so it can be sent through an AJAX function to the server, and ultimately to Python. When the ‘Post’ button is clicked, an AJAX function is executed, sending the data as a JSON to the server. On the Python side, now one must request this data that has now been uploaded to the server. Using a ‘get’ request action, the string uploaded to the server can now be used as a variable in Python.

In a next step the user must save this JSON data to a file so that Grasshopper may open it and use its location information to perform site specific analysis. The code opens a file called ‘json_test.JSON’ and re-writes its contents to the value of ‘myJson’. Therefore, every time the client clicks ‘POST’ and uploads new data to the server, the file is automatically re-written and a different analysis is performed in Grasshopper.

4.3. STEP 3: COMPUTE THE RESULT IN GRASSHOPPER

Our research has unveiled that the process of reading the JSON file in Grasshopper is quite simple. It involves using the ‘Read File’ component to read the text document, bringing the string into Grasshopper. Once the latitude and longitude coordinates are brought into Grasshopper, they need to be converted into Universal Transverse Mercator (UTM) coordinates to construct a point within Grasshopper. This is because digital workspaces such as Rhino are based off a flat infinite grid, whereas longitude and latitude are used to represent the earths surface as a spherical object. UTM coordinates are used to represent the earth in two dimensions. It is the projection system used for Google Maps and OpenStreetMap. Converting these coordinates is a complex process involving multiple calculations. Rather than trying to deconstruct these calculations we considered automated conversion systems that were readily available on the internet.

There is a variety of packaged python libraries that contain latitude / longitude to UTM conversion functions within them. ‘Pryproj 1.9.5.1’ (Whitaker 2016) and ‘utm 0.4.1’ (Bieniek 2016) are the most common packages that are readily available. By using the python libraries, the complex process of conversion can be simplified to a line of script and can be embedded within Grasshopper. Once the UTM coordinates are created they can be used as XY coordinates to construct a point in Grasshopper.

In a next step the buildings and topology of San Francisco were brought into Grasshopper using Flux Site Extractor (2016) as meshes using the Flux plugin for Grasshopper. With the point from the pipe contextualising the site, we can create multiple tools within Grasshopper, providing an automated analysis of the site. In this case, we are constructing a shadow analysis tool.

The point constructed in Grasshopper representing the users click, can be used to select the closest building footprint. This is done using Grasshopper’s native component called ‘Mesh Closest Point’ which finds the closest point projected on a mesh in relation to a search point. Using the input point from the user as the search point, the distance between the point and the surrounding building meshes can be found. The list of distances is then sorted numerically, and the shortest distance is used to define the building the user has selected. The surrounding context is chosen using the same distances, and applying a ‘Smaller Than’ component to
filter out the buildings which are outside of a variable distance. For example, if the variable is set to eighty meters, any building which is within eighty meters of the user input point will be selected.

Using the building selected by the user, and its surrounding context, one can now begin to create an analysis to feed back to display on the website. Using the EPW weather file of San Francisco, Ladybug can generate sun vectors based off a time of year and range of hours. These vectors and the building meshes selected by the user can be analysed using the ‘Ladybug_Sunlight Hour’ component, which calculates the number of hours of direct sunlight received by an input geometry. The component analyses each mesh face in an input geometry and outputs the number of hours of sunlight received each face as a list of numbers. The component also creates a coloured mesh which visualises this sunlight information. Therefore, the more mesh faces that the input geometry has, the more accurate the analysis will be. However, rebuilding the meshes with more mesh faces slow down the processes speed as each face needs to be calculated individually for sunlight access.

Once Ladybug has calculated the hour of sunlight on each mesh face, the meshes need to be transformed into a format that is suitable for web-based platforms. Wobbegong, is a plug-in for Grasshopper that is currently in development. It creates scalable vector graphic (.SVG) files using curves and polylines directly in Grasshopper, the graphics are style using cascading styling sheets (CSS) which are also defined within the Grasshopper environment.

The first step in using the API for Wobbegong, is to create a layout using the ‘SVG Plane by Page’ component which uses a plane, page size and scale as its inputs. The mesh created from the analysis needs to be oriented within this layout and all mesh faces need to be converted to polylines. Polylines can be converted into SVG scripted geometry using the Wobbegong component called ‘SVG Polyline’. The input for this component includes defining the class which associates how the polyline will be styled using CSS. The ‘CSS Style Maker’ component allows the styling of SVG based off different classes and outputs a style sheet. Options for styling include stroke thickness, colour, fill, opacity and line type. By allocating a different class to each mesh face we can associate a different fill colour in the SVG (figure 2). It’s important to note that the order in which the layers are sorted is associated with the order they are layered in the SVG. Items that appear first in a list appear underneath objects which are later on in the list.

To create a top view plan of the analysis, only the roof top mesh faces need to be visualised. This requires the need to remove any unnecessary or unseen mesh faces to display in the SVG file. To do so the centroid of each mesh face was deconstructed and sorted by height. The points which were highest in each building geometry were defined as the top faces of the geometry. Anything lower that the highest points of the geometry were culled out of the list and not used to create the plan view of the analysis. The classes used to define the mesh colours were based off the output of the ‘Ladybug_Sunlight Hour’ component which gives the hour received by each mesh face as a number. Since both the list of polylines and numbers are respectively listed. The number of sunlight hours were remapped through a gradient which converted the numbers into RGB colours.

A similar process was used to create 3D perspectives of the analysis mesh. The
mesh faces were ordered based off their closest distance to the XY plane. After the SVG and CSS are created they are used as inputs in the Wobbegong component called ‘HTML Complier’ which exports both as a single html file which can be viewed in web browsers.

![Image](image.jpg)

*Figure 2. SVG outcome of sunlight analysis in plan (left) and perspective view of Sunlight analysis in SVG (right).*

**4.4. STEP 4: SEND BACK THE RESULTS THROUGH PIPE TO URBAN PINBOARD AS RENDERED 3D OBJECTS**

When Grasshopper has finished its calculations, and has rendered out and saved a SVG file, Python must know how to look for a file and then return it to the server. A ‘while’ loop in Python uploads the rendered image to the server when that file begins to exist. If the file does not exist yet, the ‘else’ statement is executed and a placeholder image is uploaded, telling the client that no calculations have been performed yet.

At any one point, an image will be uploaded to the server. The image will change when Grasshopper completes its calculations and saves the rendered images, which will trigger Python to load that picture to the server. What SVG is uploaded to the server, however, must be received and inserted into a container for viewing for the client. The opposite of an AJAX Post, the Get function receives the SVG for use in the HTML. If the Get function is successful, the SVG is serialized to a String and inserted into the inner HTML of the ‘Content’ container. This function is wrapped in a ‘setInterval’ function, which refreshes the AJAX Get method every three seconds, allowing for the website to be updated dynamically. As Urban Pinboard was not in function at the time of submitting the paper figure 3 shows the rendered result NOT in the 3D urban model, this is what Urban Pinboard will offer when completed.
5. Significance of Research

Based on research and literature reviews, as well as through the practical experimentation within the research’s industry partner, Cox Richardson Architects Sydney, we can conclude that Urban Pinboards new transference of data between platforms has the potential to streamline and inform the industry, with huge a potential impact. This statement is based on the observation that:

Firstly, communication barriers between web-based data depositories and analytical computational design software can be broken down through new and emerging programming methods, using Javascript and Python. Individuals, such as architects and clients, with minimal or no computational design background, will be able to use this simplified interface system to access diverse processes and information, which was previously unavailable to them. Using this web-based tool in an architectural work environment would potentially cut the cost of manual analysis and economize on time. In summary, the techniques illustrated by this research have the potential to lower costs in two senses: first, it will lower the amount of expertise required to access a complex digital process and secondly, it will lower the financial cost of accessing that process.

Thirdly, due to the reduced cost in accessing design processes, Urban Pinboard has the potential to act as both - a unifying and dividing force within the architectural industry. It will unify the industry by facilitating better data sharing and widely distributing state-of-the-art processes as soon as they are written (by publicly exposing these processes as APIs). It also has the potential to divide the industry by dividing firms into smaller pieces. Lower communication costs will mean that clients can divide a design brief into portions, and give each portion to a separate firm without dramatically increasing design cost. For example, a client will likely be unwilling to pay an architect for a day's work producing a shadow study if one is able to obtain the shadow study with one click. Smaller communication overheads has further the potential to change the composition of the architectural
services market to be comprised of smaller, more specialized firms collaborating inexpensively on a shared model of a building.

Finally, changing a workflow to Urban Pinboard in combination with Grasshopper through the developed pipe can save a firm (and therefore a client) time and money on, environmental analysis, such as a shadow study. This causes design processes to be streamlined, and the architect can focus more on an informed, design-driven outcome. Since this kind of algorithmization of architectural workflows produces analysis more cheaply than other methods, it is likely that market pressures will push more firms to create these kind of workflows. In that regard, this project also demonstrates impressive scalability: the research team have further developed a data pipe-way to streamline site analysis to the click of a button at a cost of time that is not prohibitive, and could be widely adopted. There is good reason to believe that this method of data interactivity can be further developed to produce more diverse architectural analyses and processes, extending the presented case study of solar and shading. This is subject of upcoming research between the industry partner and academia.

6. Conclusion

This paper outlines the value of digitised data sets in the industry with the use of computational tools, by using an agile method to test and develop a ‘pipe,’ connecting open sourced data depositories to Grasshopper. Creating this inter-connectivity between web data sets and computational programs in a new bi-directional web-based tool has the possibility to change the industry and its practices. Increasingly visual and direct communication processors have the potential to create a large cultural impact in the interaction between the architect and client by lowering the barriers - both in terms of expertise and finance - between client and architectural process: increasingly the client communicates directly with algorithms and the architect is evermore the author of those algorithms.

The introduction of this new workflow model into the field would permit greater consideration in design driven outcomes, unlock communication barriers with technology and enable more rigorous and considered design without a significant increase in cost.

Acknowledgement

This research was enabled through the City Life Labs Grant by the Urban Development Institute of Australia (UDIA) New South Wales Chapter and developed in conjunction with Cox Richardson Architects Sydney and the Computational Design degree at UNSW as part of a Bachelor of Computational Design embedded industry thesis program.

References


Bell, G. and Dourish, P.: 2007, Yesterday’s tomorrows: notes on ubiquitous computing’s’ dominant vision, Personal and Ubiquitous Computing, 11, 133-143.


