SHAPE YOUR WORLD

Using grasshopper to create an online modelling tool for participatory design

T. WALDER,
UNSW, Sydney, Australia
1thomaswalder1@gmail.com

Abstract.

It has never been easier to collect feedback, thanks to the internet and social media. Tools to display 3d models and maps have enabled the development of online city information displays, and community consultation tools. However, most online participatory design is limited by the methods they use to involve the user in the design process. In most, input is limited to text-based commenting. Some do this in elaborate ways, providing survey tools, message boards, or by allowing comments to be located at specific points on a model or map. But despite this use of maps and models to visualise the plans to be discussed, these sites lack functions to let the user express themselves through altering the design directly. There are many reasons for this - learning to use 3d modelling tools takes training. When designing a building or place, the issues which must be considered are complex, and multiple. Design takes experience and expertise, which presents a barrier to taking part in it. Additionally, the more time and effort an online platform demands, the less likely it is to retain the attention of a user.

Computer games already have solutions to many of these problems. Most computer games have a set of rules by which the performance of the player is judged, often abstractions of real life tasks. Many computer games feature a 3d creation aspect. When a task is framed as a game, it is easier for that task to hold a person's interest. This research explores how professional computational design tools could be used, to enable gamified public participation in the design process. Using Grasshopper coding to create a backend, in such a way which can create many built forms, using inputs received from a website as JSON, and then sending the results back using scripts and formats developed by COX and UNSW CoDe. This demonstrates one application of these tools, tests their limits, and furthers the breakdown of the barriers and changes the relationship between expert designers and interested members of the public.

Keywords. Grasshopper 3d: Participatory design; gamification; community consultation;

1. Introduction: Research Aims and Motivations

Sandbox creative games, like Minecraft, SimCity, spore and Kerbal Space Program, are very popular. This genre of video game has existed for decades. These games give their players the ability to create and experiment, shaping virtual worlds. Games of this type have found many ways to give feedback to the user about their designs, and ways to enable their users to design virtual creations with minimal training.

While some of this may have been mirrored in the development of professional BIM software, the idea of using these games, which often resemble simplified, gamified, CAD programs themselves has not been taken up as a means of allowing people from outside built environment professions to be involved in the process of creating the world around them.

When designing a building or place, the issues which must be taken into account are complex, and multiple. Design professionals, with years of training and experience, an understanding of the things that go into making a building, and having the technical knowledge to make it functional, constructible, and compliant with legal and social needs.

But the physical world is shared by everyone. Sometimes, the architects and designers are working remotely, and have little familiarity, or personal connection to the location they are working on. This is why community consolation exists - to learn from the experience local people have using the built environment, and to give them the ability to have a say in how their world is shaped.

Many online community engagement tools have been and are being developed to facilitate communication between different parties with an interest in shaping the built environment. These parties include interested citizens, as well as those working in built environment professions.

For the most part, input from interested citizens limited to text-based commenting. Some do this in elaborate ways, providing survey tools, message boards, or by allowing comments to be located at specific points on a model or map. But despite this use of maps 3d models to display proposed new buildings, or other modifications to the built environment, users are not given functions to let the user express themselves through altering the design directly.

However, computer games already have solutions to many of these problems. Most computer games have a set of rules by which the performance of the player is judged, often abstractions of real life tasks. Many computer games feature a 3d creation aspect, and their designers have found ways for players to model in 3d with minimal training. When a task is framed as a game, it is easier for that task to hold a person's interest, (Lieberoth, 2015) giving a game-like community engagement tool further advantages over other methods.

Delocalised computing is a new frontier in Computational Design, where rather than performing computation on a local machine, it can be performed on a separate server, then the results sent back to the user. This way, the user does not have to have a particularly powerful machine to perform the analysis. This technique, intended for use by design firms, also has potential for use in a gamified system.
Similarly, this delocalised computing can be applied to geometry creation. Modern browser based 3d applications are capable of running basic 3d editing system, but running programs like grasshopper is still computationally intensive and shuts out many users. .

Grasshopper 3d is a visual scripting system used to create geometry and perform analysis. It has a wide range of plugins which add additional features, stuff like shadow analyses. By using Grasshopper to code the back end to the modelling tool, all these features could potentially be made available on the online platform. Additionally, it should be possible for others to expand capabilities of the modelling tool by adding their own scripts.

This paper will examine examples of community consultation websites, and what they do to collect feedback. It will briefly look at a theory of gamification, methods used for 3d modelling in video games, then detail a case study creating the back end of a video game inspired 3d modelling system.

2. Research Observations and Objectives

This project aims to develop a prototype system based on Grasshopper 3d, which could be used to create the backend for an online video game like modelling tool. This tool, if further developed could be implemented as a means for online participatory design.

It will expand on work done in this area by Cox and UNSW students, adding detail and complexity to previously simple modeling tools, featured in prototype online Computational Design platforms such as Dugong and Giraffe.

For this tool to be successful, it must be:

- Simple enough for people to use with only a few moments instruction
- Flexible enough to make a variety of simple buildings
- Possible to be expanded via the adding of extra grasshopper code.
- Designed with connection to analysis tools in mind
- Compute in a reasonable time

3. Research Questions

How can gamified public participation in the design process be enabled by professional computational design tools?

4. Methodology

This research was done as action research. The definition of action Research, as given by Rory O'Brian: "is “learning by doing” - a group of people identify a problem, do something to resolve it, see how successful their efforts were, and if not satisfied, try again" (O’Brien, R. 1998) Action research works in cycles - "Each cycle has four steps: plan, act, observe, reflect."(O’Brien, R. 1998) This action research was done with Cox Architecture as an industry partner, who gave use of some their resources and expertise and ideas from people at that company.

5. Background Research

5.1. COMMUNITY ENGAGEMENT PLATFORMS

5.1.1 *City information display websites*.

In *URBAN PINBOARD Establishing a Bi-directional Workflow Between Web-based Platforms and Computational Tools*, Johansson, Khan et al, discuss the potential of an online platform for visualizing GIS data, accessing computational design tools, and serving as communications between stakeholders in built environment. Urban Pinboard, the tool being developed, generates 3d views of buildings, roads and terrain, as well as visual representations of planning regulations. It is intended as a digital marketplace, on which others can add ideas, or develop tools to add to it. (Johanson et al., 2017)

In another paper on the same tool, Haeusler , Asher, and Booth describe how "Community Input: local knowledge and insight that already exists but is normally locked away." (Haeusler, Asher and Booth, 2017)However, currently the feedback methods in this prototype website are limited to text based communication. Urban Pinboard is intended to have different interfaces for different types of users. The experts have the ability present proposals through 3d means, but this is not available to other users. This uneven two way communication aims, this system maintains the conventional paradigm of expert designers presenting ideas to the general public, and the general public commenting on them.

Several similar online platforms exist, often with variants on the same features. Of the examples found, none feature the ability for the general public to contribute by altering 3d models.

City planner online does have functions for placing and Manipulating models, but this is only for paying accounts. - governments, developers, or architects - these tools are intended for setting up the protect models for display. (CityPlanner, 2018)

5.1.2.Gamified feedback collection

The examples above are all city information displaying websites. Other websites do community participation differently. Community planit and arki_nopoly are both examples of gamified consultation tools. Arki_nopoly is primarily an educational tool, designed to create awareness of planning processes.(arki_lab, 2018) Players answer questions - providing words based feedback in a way similar to the website based examples discussed above.

Community planit is a more active form of community engagement. Participation is a lot more involved, with the website sending users out on missions, in which they must take photos of locations, and comment on what other users of the site post. By doing this, users earn points which can be spent to bring attention to issues of their choice. (Pienaru, 2018) This is an example of a gamification strategy, making people perform tasks (gathering of information) for reward (bringing attention to issues), applied to real life. This is taps into a desire to have influence, and do civic duty.

5.1.3 Visionmaker

Visionmaker takes different approach. Users are given a map of New York city, and through a cell based system, are able to designate alternate land uses on the map, to create their own vision of how a city could be. The range of land uses a user can add is varied, including a range of ecosystems, building types, and infrastructure. The website features the ability to simulate functionality of the user's creation vision. Pressing recalculate allows the user to see how their "vision" performs in terms of water usage, carbon emissions, etc. (Visionmaker.us. 2018)

This system is simple, and easy to use, and reminiscent of the classic computer game, SimCity. While not necessarily "gamified" in the beating levels and competition against others sense, the evaluation system does provide feedback on how well a user's design performs. This system does tap into a desire to create, which creative modes in games like SimCity or Minecraft tap into. However, during use of the tool researchers found that people tended to be conservative in the types of "visions" they created. While they frame this as a bad thing, it could a result of people wanting something genuinely achievable, rather than unrealistic goals. (DuBois et al., 2017)

5.1.4. Dugong and Urban Code

Dugong was a website developed by previous UNSW honour students, another project using grasshopper to process data for online use. It uses digitised planning regulations to generate feasibility studies, displaying the results with a visual, 3d representation. Setting up a building in this system involves drawing 2d base shapes, and extruding them to create building envelopes. This is simple, yet adequate for the task at hand.

While the system was not intended as a game, it can be treated like one. The report describes one tester, who during testing succeeded in gaming the system by creating a highly convoluted base plan, inverted the base and height inputs, and tricked the system into approving a building which did not comply to regulations. This is reminiscent of a certain type of computer game player, known for exploiting glitches to beat the game. (Johanson, 2017) It is important to keep this tenancy in mind when developing a gamified system.

5.2. GAMIFICATOIN THEORIES

Lieberoth, in his 2015 paper on "Shallow Gamification" describes how many gamification efforts focus on adding game like mechanics, such as leaderboards, quests, or point scoring, to otherwise serious tasks, rather than focusing on developing a unique, comprehensive gaming experience. (Lieberoth, 2015). The study looked at a different method - framing tasks as a game. They found that simply by framing the task as a game, through use of associated visuals and props, they could achieve similar motivational effects to without the extensive use of game mechanics. (Lieberoth, 2015) According to Lieberoth, "Positive experiences in games used for serious purposes might stem from a combination of mechanics, superficial but alluring outward design, and the expectations of fun generated when people believe they are about to play a game". Setting user mindset through look and feel is key. "small mundane tasks were considered fun, because it was done in the game world"(Lieberoth, 2015)

Essentially, these findings show that it is not necessary critical to for something to have the traditional mechanics associated with a game, as long as it has the "feel" - the aesthetic, and presentation which is expected of a game.

However, a potential limitation of this study is that it was using a board game, rather than a video game. In the discussion it does consider what it takes to convince people that a computer based activity is a game, talking about the Additionally, it considered the level of enjoyment had by people who had already chosen to do the task, and less so the importance of drawing and keeping people in.

5.3. COMPUTER GAME MODELING TYPOLOGIES

There are several different methods computer games used when having a player create something in 3D. Some of them resemble CAD programs, others are more similar to Lego construction, or making models from a kit. Here are some common ways found in computer games.

5.3.1.Voxel, or Cell/tile based

In these games, the virtual environment is based on a grid, which could be 3d or 2d. Apart from a few exceptions, every object placed in the game must occupy one or more cells in this grid, and the game's file records what each cell in the game is occupied by.

Notable examples include Minecraft and the SimCity franchise. In these games, where you can place objects is limited to this grid, placement of objects in limited, restricting what can be modelled with them.

5.3.2. Parts based

In these games ,such as Kerbal Space Program, or Garry's Mod, the player is given lego-like parts to build with. Similar to above, except placement is not limited to a grid.

In some games, placement of parts is limited, with parts only able to attach to others at pre-set attachment points. Others are less restricted, using parts which can stick to each other wherever they touch. Some even allow parts to overlap, giving players the ability to create more freely, at the possible experience of realism. Placement rules are not necessarily the same for every part in the game, and the order in which parts are placed is often important.

In Kerbal space program, the player creates space vessels from from parts. A vessel file has a tree like structure, starting with the first piece a player adds. When a new part is added, the location of this new part is defined relative to the part it is attached too. Moving or deleting one part also moves or deletes the branch parts attached to it.

5.3.3 Deformable parts based

This is an evolution of the method described above, which uses a similar parts placing approach to creating models, with an additional feature in that parts don't have fixed shapes - these parts are made from deformable meshes.

An example of this is the modellers in Spore, in which most parts can be scaled up or down, and most have "handles" which a player can click and drag to stretch out the part. Many have additional handles, which control deformations specific to that part- for example, a window might let you modify the thickness of its frame, or a roof may be changed from straight to curvy.

This style of modelling is similar to ways parametric modelling is often done in grasshopper. But where in grasshopper, input to determine how a shape berries would come in the form of a slider in the visual script, where in Spore, the input is controlled via from the part's handles.

5.3.4. Lines and area based modelling.

The methods above have the user placing individual objects, usually either by clicking and dragging from a menu, or selecting then clicking to spawn it. An alternative is to have the user draw lines, or define a region to create something. This is well suited to creating walls or floors.

This kind of method is found in the building design modes of The Sims series. However, the sims uses a a composite between this and the other methods described above - working on a grid system, which even these line based elements conform to, and having parts which. The Sims, having been originally intended as a game about architecture, is an excellent example to follow.

5.4 COMPUTATIONAL DESIGN ONLINE

5.4.1 Giraffe

Giraffe was the platform this system was built with in mind. It is is a website Cox architecture has been developing, intended as part of its own own online Computational Design platforms. Giraffe has a interface similar to that of Dugong, using polylines to define the rough outlines of a building's massing, and numerical inputs to set the floors above each outline. It uses 3js to display 3d models, and runs grasshopper scrips in the back end.

5.4.2. Humpback and Redback

These are plugins designed in previous by unsw students and Cox architecture. Humpback is a tool for turning grasshopper output into GeoJSONs. It works with 2d polyline and height data, to send simple extrusions. Redback, is designed for more complex forms, and is capable of outputting 3js mesh JSON, and a new architecture specific Json format ArchiJSON.

6. Case Study

The case study involved in the development of the backend of a video game style modeling tool, using grasshopper. This is a system of grasshopper scripts which work together asynchronously to genorate a simple virtual building.

6.1 CODING STRATAGIES

6.1.1. Simulating a front end in rhino.

When connected to the front end, this system will be receiving data through a JSON file. As the front end was not implemented, and out of scope for this project, rhino geometry was used instead.

Human plugin's Dynamic pipeline was used to take geometry from rhino to grasshopper, and the types were limited to polylines and points, similar to the types of inputs already being used by the Giraffe website. For ease of testing, data was typically imputed through the Rhino instance running the individual script using it, resulting in several rhino windows being needed to model a building.

6.1.2. Game like construction

The “lines and area” and “parts based” approaches described above have been chosen as the principles for this modeling tool. Deformable parts could be implemented in a future version.

Similar to the Giraffe and UrbanCoDe precedents, this system uses closed polylines to define walls, but rather than defining the hight through numerical input, it is be controlled by the moving the polyline up or down using the mouse.

The location of a “part” is defined by a point, rotation data, and a key identifying the type of part. These parts snap on at pre-defined locations on the walls.

Similarly, placing points is used to control the types of walls and rooves generated. These points don't define the location of new geometry, but instead tell the wall or roof code what sort of wall or roof to generate, based on the wall they are attached to.

6.1.3. Use of keys

Grasshopper is not object oriented, most data management is done using structured lists. Because of the use of JSON and rhino layer data, using a list of keys to sort data was a logical solution when it came to directing different inputs to different scripts, such as sorting data to be used in different walls generation scripts.

6.1.4. Asynchronous processing.

When computing large amounts of data, Grasshopper can take several seconds to a few minutes to produce a result. To improve speed, the computation been broken up into multiple stages and threads, carried out by multiple scripts. This was implemented using a new feature in Rhino 6 which allows data to be send between multiple instances of Grasshopper via a file, in real time. Spreading the computation allowed several things to be computered simultaneously, and prevents the system from being held up by any one script. Partial results received from earlier scrips can be used elsewhere such as by analysis code, or to display in the interim, while wating for other scripts to catch up. This does result in choppy updates to the final model, with different parts updating at different times.

6.1.5. Designing the code to be expanded

Where a section of code needs to be duplicated, such as the code for retrieving exposed edges and areas of Rooves, or for setting up new Meshes, the code was placed in clearly labelled clusters, to make doing so simpler. The rest of the code was labelled by function, with notes describing how new parts could be added.

6.1.6 Experimental looping

An attempt was made to replicate the 'Snapping feature found in computer game modelers, using anemone looping, and human plugins. Location Points are inputted through rhino. Target points for them to snap to are part of the grasshopper created geometry, such as walls or rooves. It uses the dynamic pipeline to get rhino points on certain layers and stores them in grasshopper. If a location setting point is close enough to a target point, it includes the target point in the list instead. When the target point moves, it deletes the rhino points on that layer, and re bakes them, resulting in the points being recreated in their new location. This delete bake loop was triggered when a change was detected by comparing the the location points to a previous version of themselves from milliseconds previously. However, this was unreliable, and prone to errors and infinite loops. It was decided this function would be left for the front end.

6.1.7. Testing

Complex massings were put through this system to expose limitations, and where possible correct them. Unusually large and complex buildings were modelled as a stress test, and to test speed.

Dugong, Humpback, UrbanCoDe and Red back were used to demonstrate how, as Giraffe was not available. GeoJSONs describing a building modelled in Dugong was saved, imported through Humpback, exported, and successfully loaded into UrbanCoDe's map view. Similarly, Redback was used to generate 3Js mesh JSON, then displayed on an online viewer.

demonstrating this system works with the interchange tools as intended. This had to be done manually, as automated pipelines between these websites and grasshopper had not been set up.

6.2 SCRIPTS IN THE SYSTEM

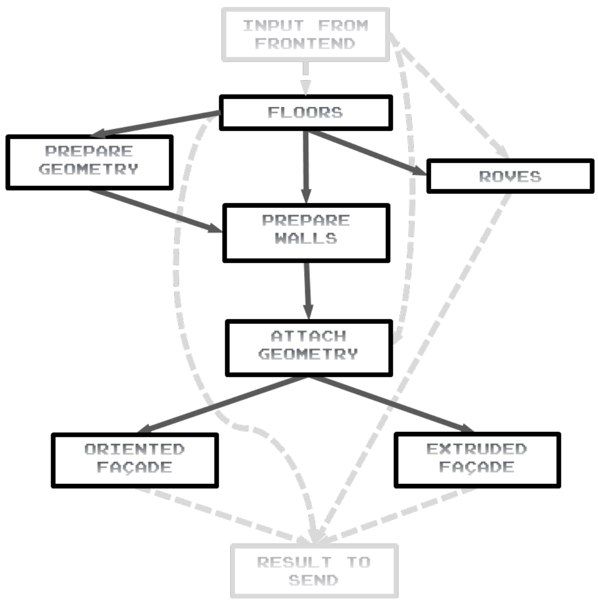

Figure 1: Script connection diagram

As illustrated in figure 1, these are the scripts which make up the prototype system. This is an overview of each. Some decisions about what functions each script performs were affected by the need to test the scripts via Rhinoceros.

6.2.1 Create basic shapes

This first script is a versatile massing creator script which also defines floor levels and outlines. The simple massing created by this first script can be used in analyses.

Multiple polylines are received from the front end (or via dynamic pipeline). Polylines must be planar. The height (z coordinate) of each is then rounded to the nearest multiple of the floor height, and the lines recreated at the new level. Then, the code checks to see if each polyline is above another, and if so, works out the height difference from the polyline to the one below. If not, height above 0 is used. These height values are used to determine how many floors are needed below each polyline.

This script also contains code which receives the points used to determine which type of wall each wall. It generates a pattern of numeric wall keys based on the keys of points attaching to the walls, or adds a 0 key if the wall does not have a point attached. Each key corresponds to a pre-set pattern of extrusions or mesh panels.

6.2.2. Rooves

The roof script receives the simple massing's and the top and bottom floors of each section. It compares the top floors with any bottom floors they are level with, to compute which parts of these top surfaces are exposed. Then it computes the parts of these surface's edges which are not touching the walls, for the automated generation of railings.

This code also receives a set of points and keys from the front end to define what types of roof should be selected. A roof is selected by proximity to a point, and that points associated key (its layer data, in the rhino prototype) determines what roof generation code it is sent to.

Then the completed roves are collected to one data stream, and sent to an exchange file, for further use or display.

6.2.3. Set up geometry

This script prepares meshes for use in other parts the code, assigning names, getting base planes, and getting the mesh dimensions which will be used to determine how much space needs to be allowed by the façade set up code if a mesh is to be used as a façade panel.

6.2.4. Facade set up

This script receives the floor outlines, and sets the lines and planes which will be used to create facades. It outputs lines, planes, and transform data for parts of the façade, which will be used later to add detail to the walls.

It uses the wall selection keys and the size data from the geometry set up script to determine how to break up the floor outlines for facade generation. For a mesh based facade, the floor outlines must be broken up into intervals the same sizes as the mesh panels to be placed there. For an extrusion based facade, the intervals are set from a list of lengths.

6.2.5. Attach geometry to facade

This script is intended to handle the addition of detail, such as doors, windows, balconies, etc, in addition to the generated facade. These additional geometries would replace a section of the facade, and this script removes the data of replaced sections, so that the façade generation scripts don't create geometry which overlaps created parts.

6.2.6 Extrusions façade

This script generates facade panels through simple extrusion. Lines received from Facade set up are extruded up by the wall height to create surfaces, then, based on a pattern of boolean values, some of those panels are extruded, resulting in sections of wall, and sections of window.

The walls this script acts upon is determined by the wall keys, it only retrieves in the numerical range that corresponds to extrude façade types.

6.2.7 Mesh facade

Like above, this code uses keys to retrieve the data needed to add the right facade to the right walls. This code uses the transform data to place Rhino 3D's blocks in place on the walls, creating a facade. The use of blocks is a stand in for adding meshes in the front end.

Because Rhino blocks don’t transfer without baking or exploding the geometry, losing their advantages in the process, this script also served as a temporary last step, receiving geometry from the other scripts to display the final, combined product.

7. Significance of Research

This research resulted in a prototype a system of grasshopper scripts which could be to be used on an online platform, for participatory design. Developing this tool shows a potential application for delocalised computing, outside of professional design contexts. It demonstrates what can and can't be done well with existing grasshopper techniques, potentially identifying gaps which could be new filled with plugins. Finally, it is another step towards opening up access to computational design tools for wider use.

Figure 1. Figure caption.

8. Evaluation of research project

This project aimed to use grasshopper to implement the backend of a video game like modelling tool. It succeeds in in the creation of a sophisticated prototyping script, which is versatile, and with some further work could be implied on an online platform.

The resulting modelling tool is flexible enough to make a variety of simple buildings, with a range of massing arrangements. It can create, and shape rooves, walls, populate those walls with geometry, and attach other geometry to parts of those walls. The code provides a framework to which further features could be added.

One limitation of this project was the online platform it was intended to work with was not available while the research was in progress, and could not be used to test it. A custom interface was not developed as part of this project - Rhino 3d's own interface stood in for a front end. While this mean that the process using this modelling tool was more complicated than could be achieved with a dedicated interface, it was able to demonstrate several basic functions that can be found in video game modelling systems.

In this project involved using grasshopper in unusual ways, and came up against some of its limits, and concluding that some parts should be left, and implemented by other means. Grasshopper lacks features that can be found in video game engines, such as detection of when an object is clicked. Grasshopper does not handle loops well. In Video game modelers, the player manipulates a file using a loop of read, change, update. Grasshopper is intended as a one way process from input to output. Those had to be left for possible future implementation in the front end, or via additional parts to the back end. Additionally, professional tools have different priorities to video games, with accurate and realistic results more important than speed. In video games, speed is important, so the user does not think the game is not working, or lose interest. This is a limitation of both nurbs modelling, and analysis tools.

8.1 Future directions

Future work could involve implementing this code on an online platform, as it was designed for. The dynamic pipeline inputs in the code would be replaced with code for receiving data from the website. Scripts could be further split, or re-merged based on how well it performs online.
Currently, adding new meshes or geometry scrips requires modification of the existing scripts in the system. Ideally, new scrips or Meshes would be added through a file system.
To gamify this system, it will need to be fully integrated with simulation and analysis tools, rather than just capable of providing suitable input to them.
On the front end, a game-like, but appropriate interface and graphics would need to be developed.
Feedback from the analysis needs to be clearly expressed clearly, simply, and visually, warning the user where they have encountered a problem.

If it is to be used as a feedback system in more than one place, there needs to be a way to configure these rules to suit different planning rules. Perhaps something similar what was developed for the Dugong website.

Alternatively, someone could look at concepts developed in the earlier sections of this paper, and develop a separate tool with a different framework, to try and meet the same needs.

9. Conclusions

This paper drew a comparison between creative computer games and the developing phenomenon of community engagement websites featuring 3d representations of their respective cities. It notes the lack of 3d modelling as a feedback method, and proposed a drawing on the ideas of video games and using professional design tools to fill this gap. In answer to this, a prototype backend for a 3d modelling system was developed using Grasshopper 3d.

Functionality of this system was demonstrated through the rhino interface, but use on an internet platform was left to possible future research.

Acknowledgements

I would like to thank my industry partner Cox Architecture, especially Andrew, Rob, Madeline, Nazmul and Emily. And from UNSW, my lecturers Nicole, Hank and Alessandra for their help and patience.

References:

arki_lab. (2018). arki_nopoly. [online] Available at: https://www.arkilab.dk/arki_nopoly-2/ [Accessed 15 Oct. 2018].

CityPlanner. (2018). CityPlanner - CityPlanner. [online] Available at: https://cityplanneronline.com/site/ [Accessed 9 Aug. 2018].

DuBois, B., Sanderson,, E., Allred, S., Giampieri, M. and Bunting-Howarth, K. (2017). maker.NYC: An Online Landscape Ecology Tool to Support Social-Ecological System Visioning and Planning. journal of Extension, [online] 55(5). Available at: https://www.researchgate.net/publication/322076747_VisionmakerNYC_An_Online_Landscape_Ecology_Tool_to_Support_Social-Ecological_System_Visioning_and_Planning [Accessed 4 Oct. 2018].

Garry's Mod (2004), Facepunch Studios

Haeusler, M., Asher, R. and Booth, L. (2017). Urban Pinboard Development of a platform to access open source data to optimise urban planning performance. ecaade, [online] 35. Available at: http://papers.cumincad.org/data/works/att/ecaade2017_011.pdf [Accessed 6 Sep. 2018].

Johanson, M. (2017). Adjudicating by Algorithm Employing open data in a computational ecosystem to inform preliminary design stages. [online] Available at: https://catalogues.be.unsw.edu.au/media/catalogue/projects/files/2017/11/22/2_Adjudicating_by_Algorithm_Employing_open_data_in_a_computational_ecosystem_to_inform_preliminary_design_stages_Madeleine_Johanson.pdf [Accessed 7 Sep. 2018].

Johanson, M., Haeusler, M., Khan, N., Butler, A. and Asher, R. (2017). Urban Pinboard - Establishing a Bi-directional Workflow Between Web-based Platforms and Computational Tools. [online] Papers.cumincad.org. Available at: http://papers.cumincad.org/cgi-bin/works/paper/caadria2017_027 [Accessed 4 Sep. 2018].
\

Kerbal Space Program (2015). Squad

Leung, E., Asher, R., Butler, A., Doherty, B., Fabbri, A., Gardener, N., Haeusler, M. (2018). REDBACK BIM: Developing 'De-Localised' Open-Source Architecture-Centric Tools. [online] Papers.cumincad.org. Available at http://papers.cumincad.org/data/works/att/caadria2018_122.pdf [Accessed 12 October]

Lieberoth, A. (2015). Shallow Gamification Testing Psychological Effects of Framing an Activity as a Game. Games and Culture, [online] 10(3), pp.230 - 244. Available at: http://journals.sagepub.com.wwwproxy1.library.unsw.edu.au/doi/pdf/10.1177/1555412014559978 [Accessed 17 Sep. 2018].

Michielsen, D., Dalle, T., Usai, M., a Romero, R. and Pak, B. (2017). Learning Participatory Urban Research Towards a Network of Collective Ingenuity (OURB). - eCAADe, [online] 1(35). Available at: http://papers.cumincad.org/data/works/att/ecaade2017_194.pdf [Accessed 8 Sep. 2018].

Minecraft. (2009, Mojang, Markus Persson, Other Ocean Interactive, 4J Studios, Microsoft Studios

Pienaru, M. (2018). The City as a Playground Game tools for interactive planning. eCAADe, [online] 2(36). Available at: http://papers.cumincad.org/data/works/att/ecaade2018_375.pdf [Accessed 14 Aug. 2018].

Spore (2008). Maxis

SimCity 4. (2003). Maxis.

Visionmaker.us. (2018). Visionmaker NYC. [online] Available at: https://visionmaker.us/nyc/# [Accessed 15 Oct. 2018].