Iterative Virtual Prototyping:

Performance Based Design Exploration

Patrick Janssen[1], Kian Wee Chen[2], Cihat Basol[3]
[1,2,3]National University of Singapore
[1]patrick@janssen.name, [2]chenkianwee@gmail.com, [3]cihatbasol@gmail.com

Abstract. *This paper proposes a digitally enhanced type of performance driven design method. In order to demonstrate this method, a design environment is presented that links the SideFx Houdini modelling and animation program to the Radiance and EnergyPlus simulation programs. This environment allows designers to explore large numbers of design variants using a partially automated iterative process of design development, design evaluation, and design feedback.*
Keywords. *Performance; iterative; prototyping; Radiance; EnergyPlus.*

INTRODUCTION

The idea of using building performance simulations to drive design decisions early in the design process has been around since the early days of CAD. As early as 1972, the ABACUS group at the University of Strathclyde built one of the first integrated performance driven design systems, known as GOAL. The designer proposes a geometry and a choice of construction materials, and GOAL then appraises the proposed design in terms of construction cost, annual energy costs, combined costs-in-use, thermal energy consumption, lighting energy consumption and planning efficiency (Maver 1970, 1972, 1998).

The authors have developed a generalised version of the GOAL type of design approach, which we refer to as Iterative Virtual Prototyping (IVP). With this approach, the designer first defines customised digital procedures for both developing and evaluating design variants. With the GOAL system, these procedures were hard-coded in the system, whereas with the IVP approach, these procedures are defined by the designer. The developmental and evaluation procedures are highly interrelated, and are together referred to as a design schema (Janssen 2004). The design schema delineates a family of possible designs that all share a certain design character, but that may vary in overall form and configuration.

Once the designer has defined their schema, then can then embark upon an open-ended exploration of this family of designs, through a cyclical process of development, evaluation, and feedback. The developmental step generates design variants which differ in their overall form and configuration; the evaluation step evaluates the performance of these variants; and in the feedback step, the results from the evaluation step are analysed and decisions are made on how to develop further variants in the next development step.

In order for the IVP approach to be feasible, a digital environment is required that will allow the designer to define their own customised developmental and evaluation procedures. These procedures may be very complex, and since most designers have only very limited programming skills, a visual approach needs to be used for defining these

procedures. Furthermore, the environment should allow for efficient and effective exploration of large numbers of design variants during the early stages of design. It is therefore critical that the environment remains agile and flexible by minimising the amount of information that is required for both the developmental and the evaluation procedures.

Based on the promising beginnings highlighted above, it might be expected that during the intervening four decades, a wide range of digital tools would have been developed that successfully integrated design development with design evaluation. Unfortunately, this is not the case. Today, there exist a massive disconnect between tools for design development and tools for design evaluation (Kolarevic and Malkawi 2005). The problem is the inability of diverse software applications to smoothly manage and exchange digital project data, which is referred to as the interoperability problem (Eastman 2008). The solution is well known, and is referred to as Building Information Modelling (BIM).

However, typical BIM solutions are not appropriate within the context of the proposed IVP design approach, for two reasons. Firstly, from a practical perspective, current BIM solutions are still incapable of supporting the smooth exchange of digital data, even between a small number of well known CAD and simulation applications. Secondly, from a conceptual perspective, current BIM solutions tend to maximise the amount of project data rather than minimize it, resulting in environments that are cumbersome and rigid. In the latter stages of the deign process, this strategy may be necessary, since a diverse set of stakeholders need to share and correlate large amounts of data. However, in the early stages of the design process, agility and flexibility are paramount. It is therefore important to use a minimal BIM approach rather than a maximal BIM approach.

This paper reports on a digital environment for IVP that successfully overcomes the two above mentioned hurdles. First, in order to allow designers to define their own customised developmental and evaluation procedures, the proposed environment uses Visual Dataflow Modelling (VDM) tools. Second, in order to allow designers to work in a flexible and agile manner, the the building information data that is being manipulated is reduced to the absolute minimum.

A DIGITAL DESIGN ENVIRONMENT FOR IVP

VDM is a procedural approach to creating design models (Woodbury 2010, Janssen and Chen 2011). It allows designers to efficiently explore alternative forms without having to manually build each different version of the design model for each scenario. Such systems are used by architects and engineers to automate design generation and accelerate the design process. Houdini is a software system that uses the procedural dataflow approach not just for modelling, but for for all tasks including animation, rendering, and compositing.

Modellig in Houdini

Modelling in Houdini consists of creating dataflow procedures. The dataflow network is created using nodes and links, where nodes can be thought of as functions that perform actions, and links connect the output of one function to the input of another function. The user visually drags nodes onto the network view from a library of available nodes. The user can then connect inputs and outputs of the nodes, thereby defining links.

Nodes have parameters that affect how the node behaves. For example, the Sphere node has parameters that define the centre point and radius of the sphere to be generated. The user may either enter the parameter value directly, or may enter a scripted expression that retrieves the parameter value from some other node in the network. This results in a second type of network, which we refer to as the parameter network.

The geometric data that nodes process has attributes associated with it. Attributes include things like x,y,z positions, normal vectors, colour values, etc. Users can view attribute data in Houdini as a spreadsheet of data. Each node in the network will create,

add to, and/or filter this data. The attribute data can be though of as flowing through the geometry network, being passed from one node to the next

Custom nodes

Houdini provides nodes for performing a wide variety of modelling tasks. However, users can also create their own custom nodes, (referred to as Digital Assets). These nodes can perform any type of task of arbitrary complexity, and they can have any number of custom parameters.

Custom nodes can be added to the Houdini environment, and used in the same way as the built in nodes. This allows for a high level of encapsulation and reuse. Custom nodes can be created to link Houdini to simulation programs. Such a custom node would first have to generate the required text based input files, then execute the program, and finally read the text based output files.

Custom nodes for Radiance and EnergyPlus have been developed. Users first create a model in Houdini using the standard built-in nodes, and then feed this model into the custom simulation node. This node then runs the simulation, and the results from the simulation are then imported back into Houdini and displayed to the user. The nodes have various parameters for setting up and controlling the simulations.

Custom attributes

In order to generate the input files, the simulation node reads the Houdini model being fed into it, and converts this model into the appropriate format for the simulation program. This is in essence an interoperability problem – the simulation node translates from the Houdini model format to the simulation model format. In order to do this, the simulation nodes need to extract the data from the Houdini model, and then restructure and reformat this data according to the requirements of each simulation program. However, this is a complex task and the Houdini model on its own does not provide sufficient information.

Entities in the Houdini model therefore need to be tagged with additional information. This is a feature that is built into the core of Houdini's approach to modelling, and consists of creating custom attributes. Some attributes, such as the x, y and z positions of points, are automatically generated by Houdini. However, users can also add their own custom attributes.

Houdini provides a set of nodes for creating, deleting, and manipulating attributes. These nodes can be used to create custom attributes for any geometric entity in the model. The data types of such attributes can be strings, integers, floats, and vectors. For example, for the surfaces in the Houdini model, the user may create a custom attribute called 'material'. Each surface in the model can then be assigned a material name.

LINKING WITH RADIANCE

Radiance is actually a collection of many separate programs that perform different tasks. The main input file for Radiance is the RAD file (and has a .rad extension). Given a RAD file, the first step is to convert this into a different file format called an octree, using a program called oconv. Using this octree file as input, three types of simulations can be performed using three different programs: rtrace, rpict, and rvu. With rtrace, the user inserts sensor points in the model and then uses the simulation to measure illuminance or irradiance at these points. With rpict, the user can generate false colour images from a particular view point in the model. With rvu, the user is presented with a graphical interface that will allow false colour images to be generated interactively, by changing the camera position and orientation. The Radiance node will allow the user to execute these programs through a simple graphical user interface.

The RAD file typically defines a list of materials and a list of geometric primitives. Each material has a name, a type, and some data that defines the properties of the material. Each primitive has a name, a type, a material name, and a set of data that defines the shape of the primitive. Primitive types for geometric entities include polygons, spheres, cones, cylinders, and meshes.

Radiance node

The Radiance node can be used to run Radiance simulations from within Houdini. The node has two inputs: one for the model geometry and one for sensor grids. The model geometry includes all the polygons to be included in the simulation.

The geometry fed into the first input on of the Radiance node needs to be constructed from polygons. The node will translate each Houdini polygon to a RAD primitive of type polygon. The polygons in the Houdini model are expected to have custom attributes to define the material. (Additional custom nodes are provided to help users define these attributes.) The node will extract the values of these attributes when generating the RAD file.

The user needs to construct the Houdini model in a way that is compatible with Radiance. Three key modelling rules therefore need to be followed. First, all polygons must be planar (although they can be any shape and can have any number of vertices). Second, polygon normals must point inwards for interior simulations, and must point outwards for exterior simulations. Third, polygons cannot have holes. This means that, in order to represent a hole in a surface, the surface needs to be modelled as a polygon that wraps around the hole.

The second input of the Radiance node is for inputting sensor grids. This is optional and is only required if an rtrace simulation is going to be performed. When the rtrace simulation is run, the simulation results will be copied to the sensor points as attributes. This then means that the results from an rtrace simulation can be graphically displayed inside Houdini, using coloured surfaces.

LINKING WITH ENERGYPLUS

EnergyPlus is an energy analysis and thermal load simulation program. Based on a user's description of a building, EnergyPlus can calculate the heating and cooling loads necessary to maintain thermal control setpoint conditions. EnergyPlus consists of a single executable that can perform many different functions. The main input file for EnergyPlus is the IDF file (and has a .idf extension). When EnergyPlus is executed, it reads the IDF file and the weather file that contains the weather data for the simulation.

The IDF file is significantly more complex than the RAD file. Firstly, the IDF file needs to specify a large number of settings for running the simulation, such as simulation parameters, location information, schedules, HVAC system details, output reports, and so forth. Second, when describing the geometry, EnergyPlus needs information about how various elements such as floors, walls and windows are related to one another. In order to define these relationships, a key concept is the zone, which is an air volume at a uniform temperature plus all the surfaces bounding or inside of that air volume. EnergyPlus calculates the energy required to maintain each zone at a specified temperature for each hour of the day.

The IDF file consists of a list of objects. Each object has a type, such as Zone, followed by a set of fields that describe the properties of that object. Some of the fields may reference other objects in the same IDF file. Through such references, a hierarchical relationship is defined between zones, building surfaces, fenestration surfaces, and shading zone objects. Each building surface object has a field that references a zone object, and each fenestration surface object and shading zone object has a field that references a building surface object.

EnergyPlus node

The EnergyPlus node can be used to run a limited range of EnergyPlus simulations from within Houdini. The node does not aim to expose all EnergyPlus functionality, but instead focuses on the types of simulations required at early concept design stages. The node has two inputs: one for zone geometry, and one for shading geometry. The zone geometry includes all the polygons that are associated with zones. The shading geometry is optional, and includes all polygons used to define surrounding buildings or other structures that shade the zones being simulated.

As with the Radiance node, the geometry fed into the first input of the EnergyPlus node needs to be constructed from polygons. The node will translate each Houdini polygon to an IDF object. The polygons in the Houdini model will need to have certain custom attributes that define the surface type and the construction. (Additional custom nodes are provided to help users define these attributes.) The three possible object types are a building surface object, a building fenestration object, and a zone shading object (for shading elements attached of the building). The node will extract the values of these attributes when generating the IDF file.

The user needs to construct the Houdini model in a way that is compatible with EnergyPlus. In this case, four key modelling rules therefore need to be followed. First, all polygons must be planar and must have a maximum of four points. Second, polygon normals must point always point outwards, towards the exterior of the zone. Third, walls, floors, or ceilings that divide adjacent zones must be modelled as two coplanar surfaces, one for each zone. Fourth, when inserting a fenestration into a surface, the surface should not have a hole cut into it. Fifth, windows with four points must be orthogonal.

Note that a polygon representing a building surface must have a consistent type and construction. In a conventional CAD mode, a designer may create a single wall surface that in some areas is an exterior wall, and in other areas is an interior wall. For EnergyPlus, this would be invalid. The surface would need to be split up into smaller surfaces, so that each surface is either all interior or all exterior.

DEMONSTRATION

In order to demonstrate the IVP approach, a scenario has been set up, whereby Houdini is used to generate and evaluate design variants for a highly simplified building type, set in a fictitious location in Singapore. The Radiance and EnergyPlus nodes are used to run simulations for both daylighting and energy.

The design is for a three storey free-standing structure. Each storey consists of a single room, 15m x 15m in plan, and 3m high. The three rooms are stacked on top of one other, and are each offset from the room below. This offset exposes certain roof areas of the room below, and these exposed areas are then glazed in order to allow daylight to enter the room. For the third room, similar glazed areas are added, even though there is no room above it. This is achieved by adding a fourth virtual room, used only for generating windows in the room below. Fig 1a, 1b, and 1c shows three parametric variants of of this building, each with different offsets. Seven parameters are required to control the offset and the rotation of the overall building.

In order to provide some kind of context, a simple environment has been created, consisting of a set of urban blocks. Each urban block has a different height, thereby creating varied shading affects from different orientations. (See Fig 1d.) Within this site, the building can be orientated in any direction.

The Houdini network

Varying any of the seven parameters can have a significant impact on the performance of the design, in terms of both daylighting and energy consumption.

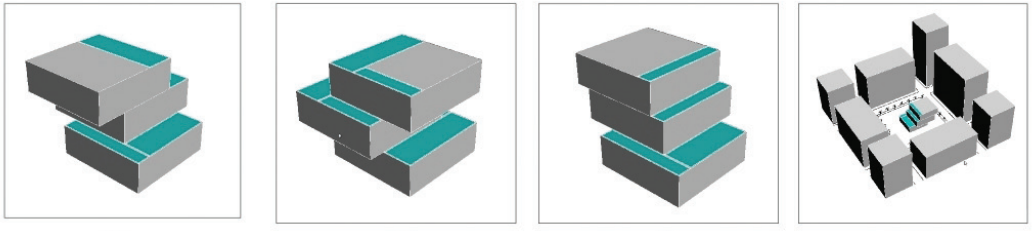
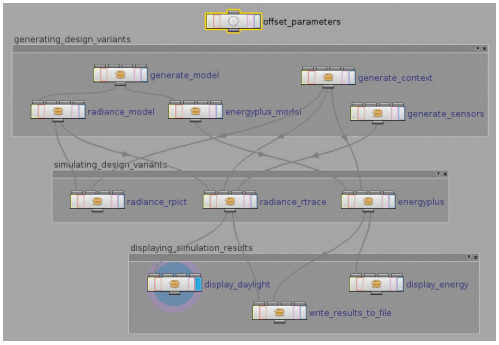


Fig. 1.
(a) (b) (c) Three design variants. (d) The design in the site context.

In order to effectively explore the impact of parameter changes, an IVP Houdini network has been created that links design generation to design evaluation. The network includes a set of sub-networks for generating all the model polygons, for running the simulations, and for displaying the results. Each sub-network has a set of nodes inside. The network is shown in Fig. 2.



At the top of the network, a node is created to store the seven parameters for generating the design variants. These parameters are connected to various nodes in other parts of the network via parameter expressions. Changing any of these parameters will trigger a new design variant to be generated, which will trigger the Radiance and EnergyPlus simulations to be executed. Once the simulations are complete, the results will immediately be displayed within Houdini, thereby giving fast feedback to the designer. In this demonstration, it takes approximately half a minute for the network to be re-executed after parameter changes have been made.

For generating the geometry, three sub-networks have been created. The generate_model sub-network creates the main design model, the generate_sensors sub-network creates sensor grids within each of the three rooms, and the generate_context sub-network creates the neighbouring urban blocks.

As well as adding attributes, the geometry also needs to be further manipulated so as to ensure that the modelling rules for each simulation program are not broken. These additional manipulations are performed inside the radiance_model and energyplus_model sub-networks. For Radiance, holes for windows have to be cut into the roof polygons using boolean operators. In addition, polygon normals have to point inwards, towards the interior of the space. For EnergyPlus, each polygon has to be split into sub-polygons, again using boolean operators. In this case, polygon normals have to point outwards towards the exterior of the space.

For running the simulations, three sub-networks have been created. The radiance_rpict sub-network runs the Radiance simulation that produces rendered and false-colour images for various viewpoints inside the rooms. (See Fig. 3a and 3b.) The images give an understanding of how light enters the room through the window slots in the ceiling. The radiance_rtrace sub-network runs the Radiance simulation that calculates daylight levels at the sensor grids within each of the three rooms. The simulations were run using an overcast sky. The data that is generated can be used to asses the design's overall performance in terms of daylighting. The energyplus sub-network runs the EnergyPlus simulation that calculates the total energy transfer for the building, and the total insolation for each surface. Three zones are created, and each zone is assigned an HVAC ideal load air system. The cooling and heating set point temperatures have been set at 25 degrees and 18 degrees respectively. The data that is generated can be used to asses the design's overall performance in terms of energy consumption.

Fig 2 (left)
Network for generating and evaluating design variants. The nodes shown in this network are container nodes that each contain another network to perform a specific task. (left)

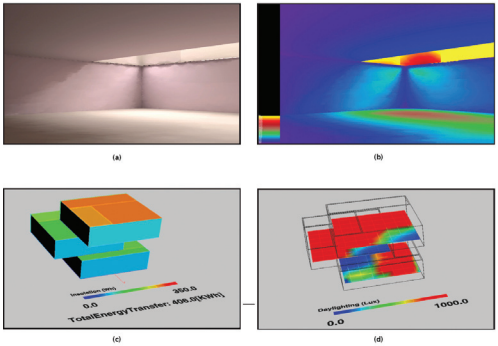


Fig 3 (right)
Simulation results for one design variant. (a) Rendering of interior space created using Radiance (rpict). (b) False colour image of interior space created using Radiance (rpict). (c) Energy transfer through each model surface calculated using EnergyPlus. (d) Daylight levels calculated using Radiance (rtrace). (right)

For displaying results within Houdini, two sub-networks have been created. In the display_daylight sub-network, the results generated by Radiance rtrace are transferred back to the points in the sensor grid, and are stored as attribute values. These values are then visualised by colouring the sensor grid. (See Fig. 3d.) In the display_energy sub-network, the results generated by EnergyPlus are transferred back to the polygons in the design model and are again stored as attribute values. The insolation data is visualised by colouring the polygons, and the total energy transfer is displayed as text. (See Fig. 3c.)

Finally, the write_results_to_file sub-network has been created for writing simulation results to a csv file. This sub-network can be used together with Houdini's animation tools in order to automatically simulate a whole sequence of design variants. This is referred to as parameter sweeping.

Parameter sweeping

Parameter sweeping involves developing a set of design variants by incrementing one parameter while keeping all other parameters constant. The variants are then all evaluated in order to understand how performance varies as the chosen parameter is 'swept' from its minimum value to its maximum value.

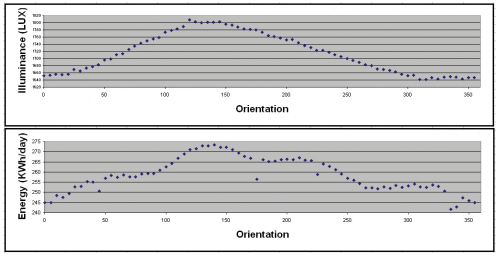


Fig 4.
Graphs showing how Total Energy Transfer and Average Illuminance change as the orientation parameter is incremented from 0 to 360 degrees.

In the case study, parameter sweeping can be demonstrated using the rotation parameter. For the offset parameters, fixed values first need to be set. The animation can then be run for a total of 72 frames, and at each frame, the rotation parameter is incremented by 5 degrees. The modified parameter triggers the generation of a new design variant, which in turn triggers the Radiance and EnergyPlus simulations. The results from the simulations are then appended to the results csv file. At the end of the animation, the csv file will contain the results for all design variants. Two graphs can then be plotted. (See Fig. 4.) The first graph plots the orientation parameter against the average daylight level, and the second graph plots the orientation parameter against the total energy transfer.

CONCLUSIONS

The research has demonstrated the feasibility of using VDM to support an IVP design approach. The proposed design environment overcomes the two key hurdles identified at the start of this paper. First, the use of VDM tools such as Houdini allow users to create sophisticated design development and design evaluation procedures. Second, the use of custom nodes allows interoperability issues with simulation software such as Radiance and EnergyPlus to be successfully overcome.

Future research will focus on developing custom nodes for a wider range of simulation programs. In particular, we are interested in structural simulations and airflow simulations.

REFERENCES

Kolarevic, B and Malkawi, A 2005, Operative Performativity (panel discussion), in *Performative architecture: beyond instrumentality*, New York : Spon Press, 2005, pp.239-246.

Eastman, C, Teicholz, P Sacks, R, and Liston, K 2008, *BIM Handbook Published*, Wiley 2008

Coenders, JL 2007, Interfacing between parametric associative and structural software, in *Proceedings of the 4th International Conference on Structural and Construction Engineering,. Xie M, Patnaikuni* I (eds.) Melbourne, Australia, 26-28 September .

Garber, R. (ed.) 2009, Closing the Gap: Information Models in Contemporary Design Practice, Volume 79, Issue 2 of *Architectural Design*, John Wiley & Sons

Hensel, M and Menges, A 2009, Patterns in Performance-Orientated Design, in *Architectural Design*, Vol 79, No 6 (November/December 2009), pp. 88 to 93.

Janssen, PHT 2004, *A design method and a computational architecture for generating and evolving building designs*. Doctoral dissertation, School of Design Hong Kong Polytechnic University (October 2004).

Janssen, PHT and Chen, KW 2011, Visual Dataflow Modelling: A Comparison of Three Systems, in *Proceedings of the CAAD Futures '11*, (to be published).

Lagios, K, Niemasz, J, and Reinhart, CF 2010, Animated Building Performance Simulation (ABPS) – Linking Rhinoceros/Grasshopper with Radiance/Daysim, in *Proceedings of SimBuild 2010*.

Maver, TW 1970, Appraisal in the Building Design Process, in Moore G T ed, *Emerging Methods in Environmental Design and Planning*, MIT Press, 1970.

Maver, TW 1972, PACE: An Interactive Package for Building Design Appraisal, in Proceedings of On-Line 72, Brunel, 1972

Maver, TW 1998, Prospects for CAAD: An optimistic perspective, in II *Seminario Iberoamericano de Grafico Digital* [SIGRADI], Mar del Plata, Argentina, pp. 6-13.

Woodbury, R 2010, *Elements of Parametric Design*, Routledge, NY.