Interactive Virtual Prototyping Architecture applied to Lighting Design: a Developmental Case-Study

Rik Beyen*
Robbie Grigg†
NHTV

Chris Damkat‡
Philips Research

Figure 1: *Graphical Illustration of virtual prototyping lighting design with proposed model.*

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities
I.3.7 [Computer Graphics]: Three-Dimentional Graphics and Realism—Virtual reality
C.2.4 [Computer-Communication Networks]: Distributed Systems—Client/server

Keywords: lighting design, pipeline architecture, virtual environments, virtual prototype, virtual prototyping, virtual simulation

Abstract- The conceptualization and communication of an interactive product plays a crucial part in the creation of a satisfactory product. Virtual prototyping helps with this by decreasing iteration time, reducing cost of prototyping, and potentially increasing the quality of the product; however, existing virtual prototyping software does not work effectively on interactive products due to the separation of virtual and physical iteration cycles. To solve this problem, a novel architectural model is proposed which incorporates these two iteration cycles in one. This is accomplished by using a client-server architecture where the server handles the hardware logic and the clients, comprised of physical and virtual clients, communicate changes to the server for synchronisation. This paper evaluates the proposed model on both work efficiency and quality in lighting design substituting a virtual client for the physical. It was found that the the proposed architecture has a 2% difference from the server output and decreases the time spent prototyping to approximately half over that of traditional methods.

*e-mail: rikbeyen@gmail.com
†e-mail: grigg.r@nhtv.nl
‡e-mail: chris.damkat@philips.com

1 Introduction

The conceptualization and communication of an interactive product is crucial to create a satisfactory product. Virtual Prototyping (VP), sometimes referred to as systems performance modeling [LaCourse 2003], allows users to implement their ideas in a Virtual Environment (VE) so that they can be used to communicate the ideas for the product efficiently, as well as being able to use the prototype as a blueprint for production. VP reduces the iteration cycles by presenting a direct visual representation of an idea without creating ambiguity in the communication of said product, which might occur in traditional methods of communicating a design such as sketches, PowerPoint presentations, and physical models [Barbieri et al. 2013; Anderson et al. 2003].

Existing VP models in the industry focus mainly on the creation of physical prototypes [Yang 2015; Blain 2014; Aromaa et al. 2014]. These often lack the ability to simulate interactions with the prototype and include a separate iteration cycle. For example: a lighting installation that changes colours depending on movement would be hard to prototype due to the interactions not being directly mappable to the real-world application; however, the ability to virtually prototype these interactive systems would provide many benefits to VP and removes the dependency of on-site hardware for prototyping.

This paper proposes an architecture that allows users to work in a VE which can simulate real-world hardware input and output from the virtual world and vice versa. The layout of this paper consists of a discussion on previous VP architectures, a proposed new architecture, an integration of this architecture in the Philips lighting design development process, and a discussion on the results from using this architecture.

2 Problem statement

Traditional production pipelines that integrate VP usually have a virtual and a physical prototyping stage, thus separating physical and virtual iteration cycles, which creates a gap in the production pipeline. As a result, the transition from a VE to a real-world environment can create complications, which is especially true when considering interactive hardware. To solve this problem an architectural model needs to be devised that does not separate virtual and physical prototyping when integrated in a production pipeline. For that purpose this paper researches if it is possible to incorporate real-time VP into a production pipeline without separating the production pipeline in virtual and physical iteration cycles.

For this research the following hypothesis were made:

H1: A lighting design production pipeline architecture can combine the virtual and physical into same iteration cycle.

H2: The proposed virtual prototyping architecture can produce results comparable to existing virtual prototyping implementations.

3 Previous work

VP is widely used in varying industries, some examples being: Ford having an immersive vehicle environment to test car parts before they are made [Blain 2014], Disney using VP for developing and prototyping new theme park attractions [Yang 2015], and Moodbuilders using VP as a service for other companies [MoodBuilders 2015]. Additionally, there are also a fair amount of companies that provide tools which allow other companies to start VP such as: Cast-Soft providing VP tools to design events and previsualize lighting designs [Cast-soft 2010], Architecture Interactive providing VP tools to support the design and construction industries with architectural visualization [WorldViz 2002], and multiple companies that provide tools for the previsualization of movies [Nvizage 2006; Take4d 2014; The Third Floor 2007]. Given these examples, however, research and documentation on VP is relatively scarce.

Robert R. Ryan [Ryan 1999] wrote a paper on Computer Aided Design (CAD) where he discussed the advantages and disadvantages of VP, focusing primarily on using VP to create car related parts. In the paper Ryan looks at how practices, such as VP, can improve the product performance and quality while decreasing the development time and costs compared to more traditional build-and-test approaches. Additionally, he examines how companies can make the transition to virtual prototype modeling by exploring industry trends related to VP and outlining the requirements for successful VP implementations. Ryan stated in his paper that:

"We may be reaching levels of diminishing return in applying CAD/CAM/CAE technologies to part design. The big opportunity to increase quality and reduce time and cost has now shifted to the system level. More significant returns on investment can be realized today through the effective use of simulation-based design processes and virtual prototyping applied to system-level design."

Ryan concluded that VP does not only reduce the number of product development iterations when compared to existing physical processes, but it is also able to increase the quality of the product.

An example of this would be the implementation of durability analysis at John Deere Welland Works [John Deere Welland Works 2006], which was a key factor in shortening the development time for its rotary cutter systems. Because they used VP over the creation of physical prototyping each test cycle was completed in one to two weeks, rather than several months. As a result of using VP, the development of a 20-foot cutter was reduced from approximately four years to one year. Ryan notes that these benefits occur only if VP is efficiently integrated in the development pipeline. A key to this is the speed at which the iteration process operates. This means that critical decisions need to be made quickly, which can be hard because people and departments are, by tradition, not integrated. Additionally, Ryan mentions that current hardware is unable to simulate real world geometry and conditions, which results in VP being an approximation. Though VP might be an approximation of the real world, a quick and approximate analysis can often highlight design flaws that might otherwise be overlooked. Furthermore, VP becomes more cost effective when products require millions to bring to the market, such as complex lighting installations.

Even though the main benefits of VP before physically prototyping are generally considered the reduced time-to-market, reduced costs, knowledge sharing and user participation [Aromaa 2013; Aromaa et al. 2012], this is not always applicable when considering lighting design due to the inability to guarantee correct mapping from VR to the real-world. In order to have effective VP with regards to lighting design it is required to directly map the interactions in a VE to the real-world. This is supported by experienced practitioners in the field of VR, who indicated that in order to work effectively in a VE, the application content must include the ability to access or change environmental/system/meta parameters, create and manipulate particular objects, perform analyses, and export changes to permanent storage [Sowizral et al. 1995]. Additionally, this is also supported by Ma et al. [Ma et al. 2011] and Bordegoni et al. [Bordegoni et al. 2009], who claim that VP is particularly useful in the assessment of interaction systems used by users.

Aromaa et al. [Aromaa et al. 2014] presented a report that concludes from VP research carried out during a large scale project called LEFA "New Generation Human-Centered Design Simulators for Life Cycle Efficient Mobile Machines". In this paper they

also propose a VP framework where the main theory applied being Engeström's activity theory [Engeström 1987; Engeström 2000; Engeström and Toiviainen 2011] and the structure is based on the human, interface, and system model elements. Due to this setup the interaction between humans and the system model have an interface layer between them, such as a VE. The primary focus of the paper is in the area of human-machine interaction design and covers: a proposed framework for VP in human-machine interaction design to be able to systematically construct and test VP, the benefits that are acknowledged to have come from VP, the implementation of VP in companies, and the application of VP during the design review.

Three distinct conclusions were made from the LEFA project when evaluating the benefits of VP for company/business, managers/designers, and users/developers. Companies can benefit from virtual prototyping in terms of reduced costs, time-to-market and number of physical prototypes and increased productivity, quality, and customer satisfaction. Managers and designers benefit from VP because processes are more efficient, sharing of information is easier, and management gets more efficient. Additionally, it can enhance the designer's experience, and improve decision-making and early design fault recognition. Furthermore, virtual prototyping is a safe environment to test critical tasks or to illustrate futuristic concept ideas that do not exist yet. VP users/operators can participate and validate their product design in the initial phases. This makes it possible to achieve better products and user acceptance, due to it being easier for the user to interact with and test prototypes.

Drettakis et al. [Drettakis et al. 2007] wrote a paper where they presented a user-centered design approach for the development of VE in real-world architecture and urban planning by utilizing an iterative and user-informed process throughout the design and development cycle. In this paper they argue for a combined approach of lab experiments and in-situ real-world usage to evaluate real-world applications. To accomplish this, a preliminary survey was done with end-users related to real-world architectural and urban planning projects followed by a study of the traditional workflow employed. Afterwards, the elements that would make VE useful in a real-world setting were determined by choosing the appropriate graphical and auditory techniques to accomplish a high level of realism. Furthermore, the evaluation of this system was done in both a laboratory and the users' natural work environments. The evaluation of the user-centered design approach suggests that involving users and designers from the beginning improves the effectiveness of the VE in the context of an urban planning project. They found several important aspects that enhance the design process and communication about designs of the VE: the sense of scale given by the combination of realistic vegetation and human figures/crowds, use of shadows, and 3D spatialized audio. Additionally, in terms of VR capabilities, multiple views were considered useful as well.

VP models that include a client-server network architecture exist as well, however, these are often focused on the accessibility of VP rather than connecting hardware and software applications. An example of this would be a patent from Gilray Densham [Densham 2013], which covers systems and methods that allow a user to visualize and customize 3D models of a venue. To do this a data abstraction of the 3D venue model is created and sent to the venue operator, which can be used to reconstruct the model in a 3D virtual environment. In practice this means that the user can modify the venue model and send the changes to a server, which will validate the received data. This method can provide many benefits with regards to the production pipeline of product design, including lighting design, but does not solve the problem of separate virtual and physical iteration cycles.

To summarize, it was found that adding VP iteration cycles to the production pipeline provides multiple benefits over traditional

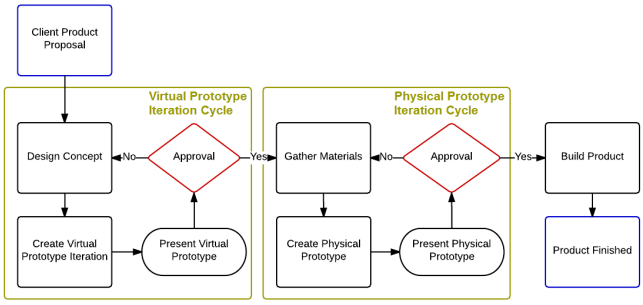

Figure 2: *High level overview of a general product pipeline that incorporates virtual prototyping.*

production pipelines when integrated correctly, including reduced costs and time spent on a project, but also an increase in quality and performance. A VP iteration cycle is usually added to the production pipeline as an isolated module, which might introduce a new bottleneck with regards to interactive products, such as interactive lighting installations. In these conditions it might be more desirable to merge the VP iteration cycle and physical iteration cycle. This would mitigate potential errors when mapping the virtual product to the physical product, which can lead to reduced cost and time spent on a project.

4 Proposed Virtual Prototyping Architecture

For the creation of a product there are many different production pipelines that incorporate VP; however, when creating a physical product these production pipelines usually have a separated iteration cycle for virtual and physical prototyping [Ryan 1999; Barbieri et al. 2013; Aromaa 2013; LaCourse 2003] (see Figure 2). The virtual iteration cycle usually consists of creating a design concept, implementing a design in a VE, and presenting the design for approval. Physical iteration cycles are usually more diverse, because different methods are used depending on the type of product; however, in its most simplistic form the cycle consists of gathering materials, creating a physical prototype, and presenting the physical prototype for approval. With regards to lighting design there can be many more factors for example: creating physical prototypes of complex lighting installations often includes traveling to the location due to lighting having a high dependency on material types and scale. When integrating VP in a production pipeline as an isolated module it is hard, if not impossible, to create a product pipeline that guarantees a streamlined process between virtual and physical prototyping iteration cycles. In general when a product pipeline is established, VP will reduce the amount of physical prototypes needed, but still depend heavily on a physical prototype, for example when evaluating haptic feedback [Ferrise et al. 2013]. This holds especially true when considering products such as complex lighting structures, which can be dependent on human interaction.

In order to bypass the potential overhead with regards to the transition between virtual and physical iteration cycles, this paper proposes a model that merges virtual and physical prototyping. To combine both iteration cycles it is crucial to calibrate the simulated hardware input and output in the virtual world, such that it properly reflects how user interactions with hardware take place in the real-world. When considering this as an additional step, the prototype iteration cycle would include: creating a design concept, gathering materials, implementing design in a VE, calibrating hardware components, and evaluating the design for approval (see Figure 3). Even though this model introduces the overhead of calibrating hardware components, it also removes the need for a separate physical proto-

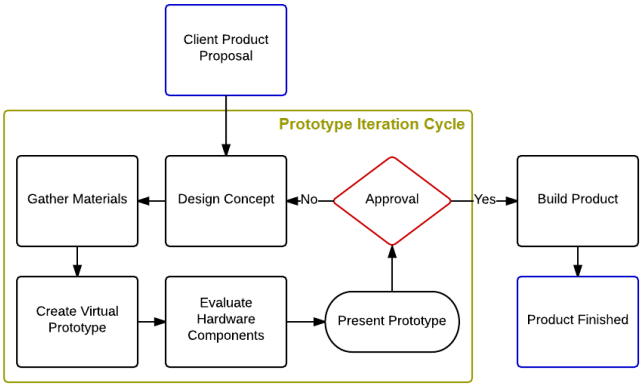

Figure 3: *High level overview of the proposed product pipeline that incorporates interactive virtual prototyping.*

type iteration cycle. As a result, the proposed model exhibits less traits from a sequential design process, which makes it more flexible compared to previously discussed traditional product pipelines.

Traditional lighting design pipelines often include large installations in places that are crowded, which can lead to complications when building these installations on location; however, these physical prototypes are crucial to find how the light interacts with the environment and how it is experienced by the observer. Other approaches, such as miniature scale prototypes, often do the same as VP does when integrated as an isolated module and therefor have the same limitations. When considering VP for lighting design, especially when interactive, the proposed production pipeline can provide some interesting benefits. By integrating the hardware and software applications in one coherent production pipeline it is possible to remotely test individual components in both the real world and the virtual world. When considering the previous example of complex lighting structures, it is possible to simulate how a changing environment impacts the lighting and how different amount of people are handled by a motion sensor that activates lights.

5 Client-Server Architecture

In order to realize the benefits of VP for the conceptualization of lighting design covered in previous chapters, it is necessary to have an architecture that is able to simulate sensory data, interpret sensory data, and distribute lighting information to hardware and software applications. This paper presents an architecture that allows for existing rendering engines, such as the Unreal 4 Engine, to be integrated in existing production pipelines and used for VP of lighting design. This prevents the production pipeline from vendor lock-in and thus increases the flexibility of the proposed model.

The implementation of the proposed architecture is integrated with Philips' existing production pipeline for lighting design, which already has a server in place to handle hardware input and output. In order to prototype efficiently it is required to have a means of receiving and modifying the state of the system in real-time with both hardware and software interaction. To enable Philips' production pipeline with real-time interaction between virtual and real-world environments a two-way interaction is made through the standardized Real Time Streaming Protocol (RTSP) [Schulzrinne et al. 1998] and Hypertext Transfer Protocol (HTTP) [Fielding et al. 1999]. The server broadcasts a continuous stream of data through a RTSP, which is best visualized as a stream of pixels, and is picked up by the real-world hardware or the client that connects the server to the render engine, in our case Unreal 4. This creates a continuous stream of information that allows the server to communicate hardware output to a VE or real-world hardware. Additionally, it is possible to simulate sensory input by interacting with the VE, which sends HTTP messages to update the server. For example: when a light switch is pressed in the virtual world, a signal can be sent by the plugin to the server. This signal is then processed by the server in the same way as a signal sent from hardware. This bidirectional interaction between real and virtual information makes it is possible to simulate both the result of actions, such as lights going on, as well as simulating the actions themselves, such as walking through a motion sensor (see Figure 4). Furthermore, the proposed architecture allows for multiple clients to connect to the server; however, the act of network data synchronisation is beyond the scope of this paper and will be left for future research.

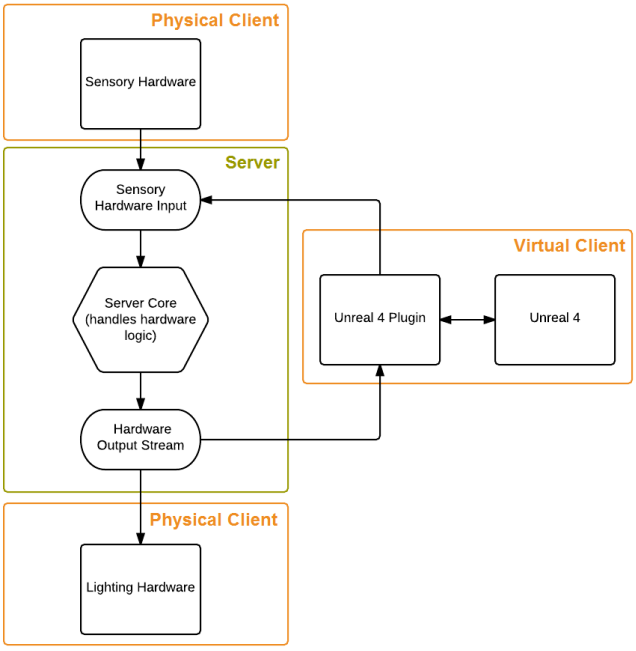

Figure 4: *High level illustration regarding the implementation of the proposed architecture.*

5.1 Server

The server has three roles: receiving input signals, updating the content, and sending output signals. As mentioned in Section 5, the server receives HTTP requests from either hardware or software applications. These requests can modify the system to a certain extent, which is predefined by the server. When the server recognizes a signal it is processed with the corresponding logic. An example of the server receiving a request is when someone stands on a pressure sensor. The pressure sensor will send out a HTTP request stating that it is pressed and possibly how far it is pressed. In turn the server will execute the logic present for such a request and updates the stream output accordingly, such as changing the colour of a light.

The server logic is considered an isolated module with a continuous output and accepts input to change the state of the program, which makes it flexible enough to be customized for ones needs without introducing changes to the overall architecture. In the implementation of the proposed architecture the server runs an algorithm that dictates the base colours of the lighting in the scene and on top of that acts as a state machine that generates different lighting output

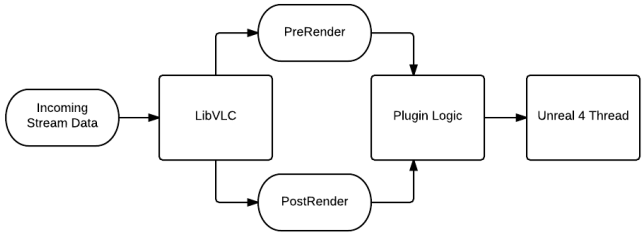

Figure 5: *High level illustration regarding the procedure of handling incoming streams for the Unreal 4 plugin.*

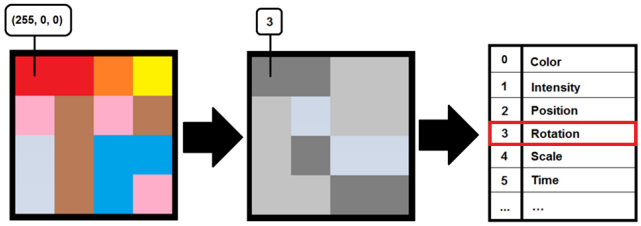

Figure 6: *Process of interpreting stream information (left) applying the mapping texture (middle) to get the context from a table (right).*

based on the received requests. When the server is finished doing the lighting computations, the RTSP output stream will be updated.

5.2 Client Plugin

The server and client use standardized protocols to communicate between each other; however, the information stream needs to be decoded on the client side. Server logic in the implementation of the proposed architecture is handled by the server that is already present in Philips' production pipeline for lighting design, which is discussed in Section 5.1. The client is required to intercept the output stream from the server in order to update the lighting of the VE. The external library LibVLC is used in the implementation to intercept the server stream. LibVLC can intercept the output stream and spawns a *PreRender* and *PostRender* event, which allows the client to fill up a video buffer. The PreRender function allocates the size of the incoming stream, sets the location of the buffer that needs to be filled, and locks the buffer until the stream has been processed. Afterwards, the PostRender function unlocks the buffer and allows the data to be read by the plugin (see Figure 5).

Information gathered from the output stream does not have any meaning by itself. To provide the received information with context an eight bit texture is synchronised between the client and server, which will be referred to as the mapping texture. The mapping texture has the same dimensions as the incoming stream, but has predefined values that map to a table in order to provide context to the incoming stream. The table contains a set of predefined definitions regarding the context of incoming data, which allows the incoming stream to be interpreted. This allows the mapping texture to take a specific index of the stream, extract the mapping texture value with the same index, and map the resulting eight bit value to a predefined table (see Figure 6). The advantages of mapping the meaning of pixels though a mapping texture is that there is no header overhead in the stream, and allows for a more flexible grouping of data.

The plugin can simulate real-world interactions in a VE by sending HTTP requests to the server, which removes the overhead of a bidirectional stream connection. For this approach no additional third party software was used in the plugin, because Unreal 4 has native

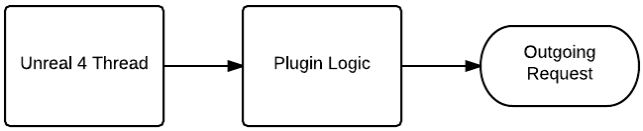

Figure 7: *High level illustration regarding the procedure of handling outgoing requests from the Unreal 4 plugin.*

support for HTTP requests (see Figure 7). Hardware and software applications use the same protocol to communicate to the server in the proposed architecture, which allows the server to treat incoming requests the same regardless of the sender. This makes it possible to simulate requests from physical hardware with requests made from a VE (see Figure 4).

5.3 Client Interaction

Client interaction is often heavily dependent on the type of rendering engine that is used for VP. The Unreal 4 Blueprint system is used in the implementation of the proposed architecture, which allows for the creation and modification of lighting by both artists and designers [Golding 2014]. The Blueprint system in Unreal 4 is a visual scripting system based on the concept of using a node-based interface that allows users to use a wide range of concepts and tools generally only available to programmers.

When the plugin receives data from the PostRender function it is copied to the rendering buffer present in Unreal 4. This buffer can be accessed through any C++ or Blueprint class, which allows for a flexible interaction with regards to both traditional code and visual scripting. Likewise, sending a signal to simulate hardware input can also be done through C++ and Blueprint classes. The custom Blueprint classes made for the implementation can be used as visual scripting nodes and essentially enables the user to do three things: retrieve data from the server, interpret data from the server, and send requests to the server. Additionally, these visual nodes can make networks of logic that manipulate the received data such that it is possible to seamlessly connect them with other visual nodes present in the Unreal 4 Engine, which can then be used to enable interactive VP (see Figure 8).

6 Evaluation

Several usability tests are done with a prototype using artists and designers from both Philips and MoodBuilders to evaluate if the proposed architecture is useful for lighting design. These participant are familiar with Unreal 4 and will do three survey based evaluations for different iteration cycles in a random order, where each task to accomplish is also randomized. These iteration cycles include: traditional product pipeline iteration cycles with a perfect transition, traditional product pipeline iteration cycles with an imperfect transition, and the proposed product pipeline iteration cycle. Additionally, the amount of time they spend working is recorded in order to find out which iteration cycle was the fastest. Due to time constraints and lack of access to physical systems, the physical prototype iteration cycle is simulated by a reimplementation of the previous virtual prototype iteration cycle in a VE. This will bias the evaluation generally in favour of the traditional pipelines; however, the evaluation could still give valuable insight if the proposed architecture is a concept worth proceeding. The process of the iteration cycle evaluation is noted in Section 6.1. Afterwards the calibration accuracy of the proposed pipeline, results of the survey evaluation, and time working on different iteration cycles are presented in Section 6.2.

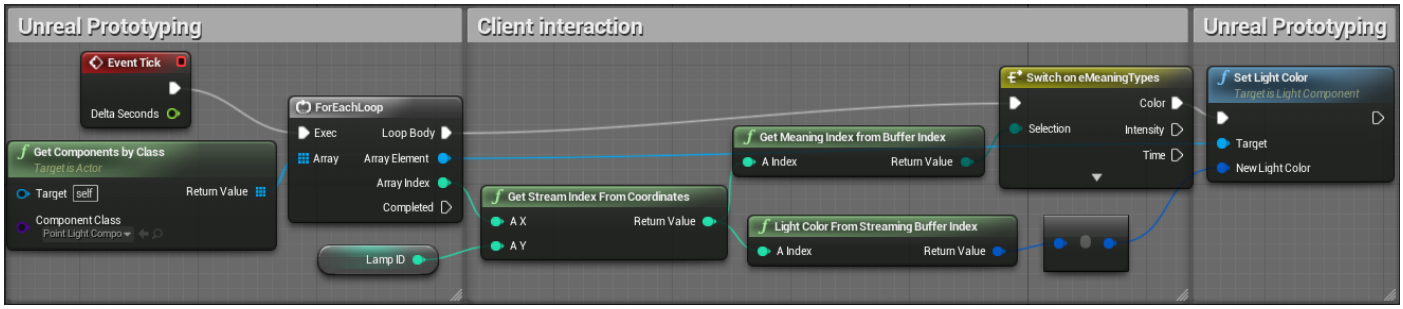

Figure 8: *Illustration of a basic Unreal 4 Blueprint implementation for a lamp.*

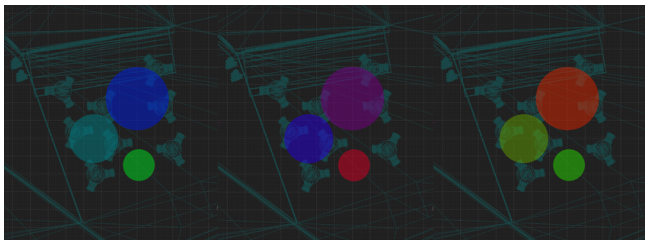

Figure 9: *Illustration of top-down overviews given to the participants.*

Survey #0	Grade	Additional information to the statement (optional)
I was able to create the lighting quickly.		
I am happy with the quality of the lighting I was able to achieve		
Minimal complications were present when creating the lighting		
I liked the overall experience of this way of working		
This method is great for lighting design		
This method is great for other products/designs		

Table 1: *Illustration of the survey, where each statement is graded between 1 (complete disagreement) and 9 (complete agreement).*

6.1 Methods

The evaluation procedure for each participants will follow the following schedule:

1. Inform the participant that they need to create lighting for a restaurant three times.
2. Make it clear that the systems are being tested not the participant and encourage them to talk out loud as much as possible.
3. Provide the participant with a random top-down overview of the restaurant and the lighting that needs to be created (see Figure 9).
4. Inform the participant that this overview symbolizes the wishes of the restaurant owner with regards to lighting and that they need to create it.
5. Introduce the participant to one of the following pipelines randomly.
   (a) Create the lighting in the Unreal 4 scene normally (see Figure 10), then recreate it in the same scene using the previous iteration as reference, simulating the best case scenario of a traditional VP production pipeline iteration cycles.
   (b) Create the lighting in the Unreal 4 scene normally (see Figure 10), then recreate it in a scene with a different light setting (see Figure 11) using the previous iteration as reference, simulating a problematic scenario of a traditional VP production pipeline iteration cycles.
   (c) Create the lighting in the Unreal 4 scene with the proposed client-server architecture, simulating the proposed production pipeline iteration cycle.
6. Introduce the allowed Unreal 4 controls and how they work.
7. Let the participant implement the lighting from the provided overview (point 3) in the provided production pipeline (point 5).
8. Record the time spent working.
9. Let the participant fill in the survey when the task is completed (see Table 1).
10. Repeat from point 3 with new randomly provided information for point 3 and 5 until all scenarios have been used.

6.2 Results

The merging of virtual and physical iteration cycles in a lighting design production pipeline (H1) was successfully implemented with the proposed client-server model. As a result it has become possible to create and prototype individual modules within a larger project. An example of this would be calibrating a motion sensor in a lab and using the VE to simulate a crowd interacting with the sensor in the virtual world. Additionally, the architecture of the proposed model is flexible enough to switch out either the server or client when needed, due to the standardized communication protocols used for data transport.

In the implementation of the proposed architecture, the server is based upon Philips' existing server and it is assumed that the data sent by the server correctly represents hardware lighting output. The lighting recreated on the client displays colours with more than 98% accuracy for each colour channel when calibrated (see Figure 12 & 13), which makes it indistinguishable from the lighting generated on the server (H2).

In the results of the survey it is fairly unanimous that a traditional pipeline with an imperfect transition is inferior to the other two

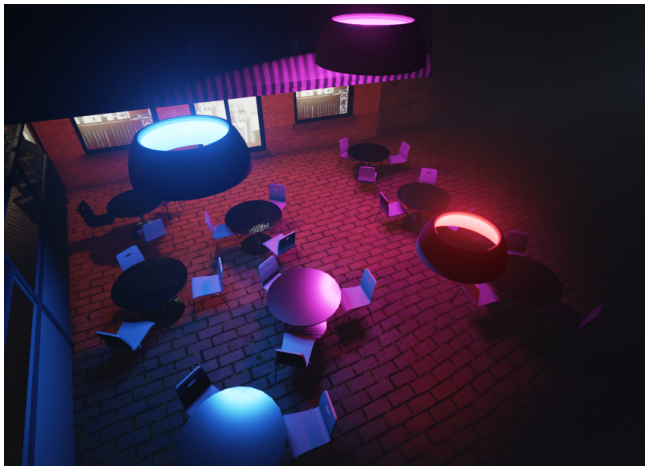

Figure 10: *Example of a participant's first iteration cycle in the traditional pipeline, which is used to create the second iteration cycle.*

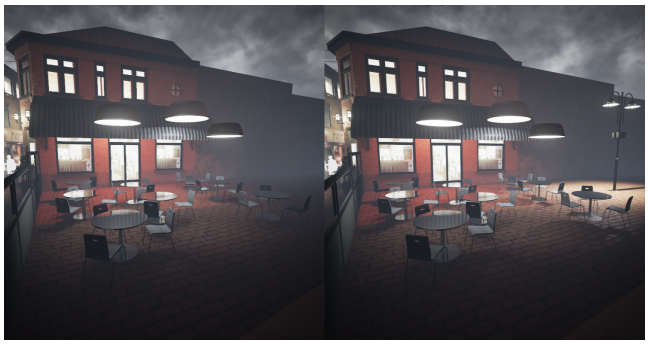

Figure 11: *Difference between the base scene (left) and the scene with an additional lantern (right) used for an imperfect transition.*

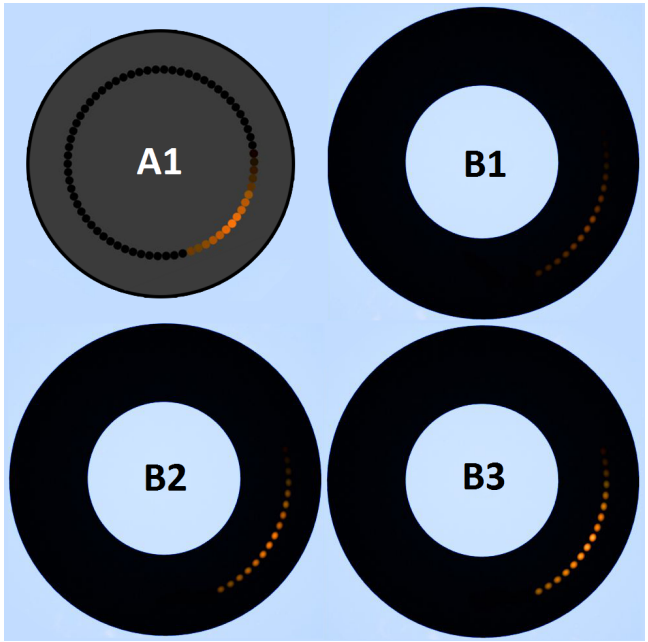

Figure 12: *Light comparison between the light colours generated on the server (A1) and the 3D lights generated by the client with 1000 (B1), 4000 (B2), and 8000 (B3) units of intensity.*

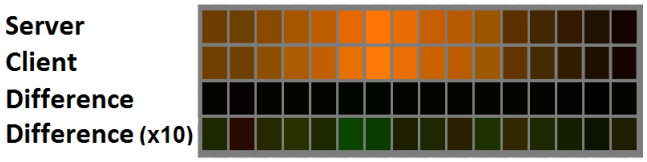

Figure 13: *Light colours of the server, client, and the difference between them when calibrated.*

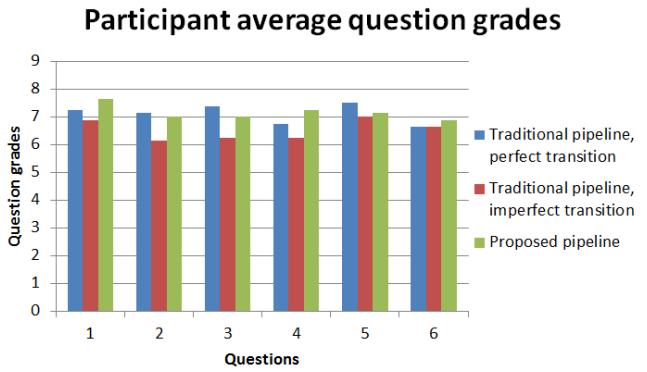

Figure 14: *Overview of the rating of each question regarding the iteration cycles.*

pipelines in almost all categories; however, the rated score between the proposed pipeline and traditional pipeline with a perfect transition changes based on the context (see Figure 14). The participants liked the lighting design speed and the overall working experience of the proposed pipeline more than the traditional pipeline with perfect transition, but also mentioned that there were more complications when working with the proposed pipeline. This is in agreement with the time recordings of the participants where the traditional system with perfect transition had an average time of 319 seconds, the traditional system with imperfect transition had an average time of 350 seconds, and the proposed system had an average time of 163 seconds (see Figure 15), which makes the overall iteration cycle of the proposed pipeline approximately half over that of the traditional pipelines. Additionally, the participants perceived an average difference in quality of approximately 2% between the traditional pipeline with perfect transition and the proposed pipeline, which is in agreement with earlier results that showed that calibrated lights on the client have a 98% accuracy when compared to the lights generated on the server.

7 Discussion

It is possible to merge the physical and virtual iteration cycles, as mentioned in Section 6.2, which can provide several advantages over the traditional method of integrating VP in a production pipeline. One of these advantages is a single iteration cycle which mitigates the problems that might occur when moving from a virtual iteration cycle to a physical iteration cycle. The participants worked roughly 6 seconds slower on average between the proposed iteration cycle and the first iteration cycle of the traditional pipeline with a perfect transition (see Figure 15), which means that the over-head introduced is fairly low and does not, unless new hardware is used, have to be redone for each iteration cycle. This makes the earlier mentioned overhead of the proposed production pipeline negligible in comparison to the full iteration cycles of traditional production pipelines. Additionally, the proposed model allows for hybrid evaluation of multiple scenarios, which can be much more

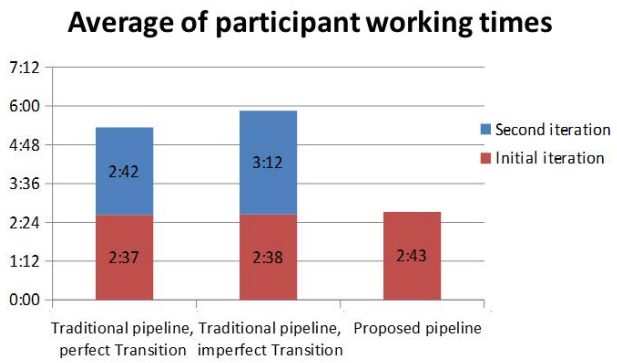

Figure 15: *Overview of the time spend on each iteration cycle.*

accurate than an evaluation with traditional VP. An example of this would be evaluating how lights react when connected to a camera or detailed motion sensor that form lighting patterns based on crowd interactions.

Even though the proposed pipeline has many benefits for lighting design, it does have some disadvantages when put in a different context. When considering non-interactive products the benefits of the proposed model with regards to testing interaction, such as hybrid testing with individual components, are not applicable anymore. Additionally, there are many stand-alone software applications already available for the integration of traditional VP [WorldViz 2002; Cast-soft 2010; Nvizage 2006; Take4d 2014; The Third Floor 2007], which are not directly compatible with the proposed architecture. Finally, not all hardware is capable to connect with a client-server architecture, which can make the proposed architecture impractical for some products. Due to these advantages and disadvantages the proposed architecture does not replace traditional VP integration, but rather offers a novel client-server architecture that is able to improve upon the production pipelines that are compatible, such as a production pipeline for lighting design.

For lighting design specifically, this model provides a novel way of iterating on the product and eliminates the bridge between virtual and real-world prototyping. In Section 6.2 it was found that, on average, the participants had less complications when designing lighting in the traditional pipeline with a perfect transition than with the proposed pipeline; however, the proposed pipeline did enhance the production speed and overall working experience. The proposed architecture has a different workflow which could explain the larger amount of complications present when evaluating the proposed implementation. Nonetheless, it can be deduced that the proposed pipeline provides a significant boost to iteration speed, especially when considering that the proposed pipeline is able to outperform the traditional pipeline with a perfect transition.

8 Conclusion

In conclusion, creating a virtual prototype before developing a product is a widely used procedure to improve the production pipeline iteration cycles. It was found that VP is usually implemented as an isolated module, which could result in a new bottleneck with regards to interactive products. This paper proposes a novel architecture for production pipelines that uses a single iteration cycle to combine the virtual and physical iteration cycles, which are often present in a production pipeline that integrates VP.

It was found that the proposed implementation worked quite well for hardware that is able to communicate with a server. To illustrate this, an implementation of the proposed architecture was made in Philips' lighting design production pipeline. This implementation showed that both the server and client are able connect with each other through standardized protocols, and that the reconstruction of lighting on the client displays colours with more than 98% accuracy when calibrated (see Figure 13), which was in agreement with the results in Section 6.2. Furthermore, earlier concerns of introducing additional overhead due to the calibration needed between hardware and software applications seemed to be unnecessary as the overall iteration cycle of the proposed pipeline is significantly shorter than the traditional VP pipeline with both imperfect and perfect transitions.

Even though this model might not be suitable to replace traditional VP integration for all types of production pipelines, as mentioned in Section 7, it is found that the proposed model does provide a novel way to increase iteration speed by seamlessly iterating over the virtual and physical iteration cycles at the same time. This enhances the production pipeline for specific implementations, such as lighting design, and is validated by the evaluation by an implementation of the proposed architecture in Philips' production pipeline for lighting design.

9 Future research

The amount of participants in the evaluation of the proposed architecture has a fairly small sample size and simulates the physical iteration cycles due to time constraints. For future research a larger and more elaborate project could give more insight on the advantages and disadvantages of the proposed pipeline. Additionally, this paper attempts to research the validity of the proposed iteration cycle, which covers both physical and virtual iteration cycles in a production pipeline; however, the effectiveness compared to established production pipelines [Drettakis et al. 2007; Aromaa et al. 2014; Aromaa 2013; Blain 2014] has not been measured. For future research it will be useful to do a larger and more elaborate study that reports the advantages and disadvantages of the proposed pipeline over the course of one or multiple iteration cycles, and compare it with production pipelines used in the industry with regards to costs, time spent, quality, and user satisfaction.

In Section 5 it was mentioned that multiple clients could connect to the server at the same time with the proposed architecture; however, the act of network data synchronisation was beyond the scope of this paper. For future research it might be worthwhile to look at methods and protocols that allow for multiple clients to interact with each other, possibly increasing the iteration speed of a production pipeline even further.

References

ANDERSON, L., ESSER, J., AND INTERRANTE, V. 2003. A virtual environment for conceptual design in architecture. *Proceedings of the workshop on Virtual environments 2003 - EGVE '03*, 57–63.

AROMAA, S., S., P. L., VIITANIEMI, J., JOKINEN, L., AND KIVIRANTA, S. 2012. Benefits of the use of Virtual Environments in product design review meeting. In *Proceedings of DESIGN 2012, the 12th International Design Conference, Dubrovnik, Croatia*, 355–364.

AROMAA, S., LEINO, S.-P., AND VIITANIEM, J., 2014. Virtual prototyping in human-machine interaction design.

AROMAA, S. 2013. Designing user experience for the machine cabin of the future. In *eEngineering 2009—2012*, K. Belloni

and O. Ventä, Eds. VTT Technical Research Centre of Finland, 45–54.

BARBIERI, L., ANGILICA, A., BRUNO, F., AND MUZZUPAPPA, M. 2013. Mixed prototyping with configurable physical archetype for usability evaluation of product interfaces. *Computers in Industry 64*, 3, 310–323.

BLAIN, L. 2014. Ford demonstrates VR prototyping in Immersive Vehicle Environment. *Gizmag* (Sept.).

BORDEGONI, M., CUGINI, U., CARUSO, G., AND POLISTINA, S. 2009. Mixed prototyping for product assessment: a reference framework. *International Journal on In- teractive Design and Manufacturing 3*, 3, 177–187.

CAST-SOFT, 2010. The Industry's intuitive 3D event design, planning, and sales tool! http://cast-soft.com/.

DENSHAM, G., 2013. System and Method for Visualizing a Virtual Environment Online. US Patent 20130135303 A1.

DRETTAKIS, G., ROUSSOU, M., RECHE, A., AND TSINGOS, N. 2007. Design and Evaluation of a Real-World Virtual Environment for Architecture and Urban Planning. *Presence: Teleoperators and Virtual Environments 16*, 3, 318–332.

ENGESTRÖM, Y., AND TOIVIAINEN, H. 2011. Co-configurational design of learning instrumentalities. In *Learning Across Sites - New Tools, Infrastructures and Practices*, S. Ludvigsen, A. Lund, I. Rasmussen, and R. Säljö, Eds. 33–52.

ENGESTRÖM, Y. 1987. Learning by Expanding. *Helsinki: Orienta-Konsultit Oy*.

ENGESTRÖM, Y. 2000. Activity theory as a framework for analyzing and redesigning work. *Ergonomics 43*, 7, 960–974.

FERRISE, F., BORDEGONI, M., AND CUGINI, U. 2013. Interactive Virtual Prototypes for Testing the Interaction with new Products. *Computer-Aided Design and Applications 10*, 3, 515–525.

FIELDING, R. T., GETTYS, J., MOGUL, J. C., NIELSEN, H. F., MASINTER, L., LEACH, P. J., AND BERNERS-LEE, 1999. Hypertext Transfer Protocol – HTTP/1.1. https://tools.ietf.org/html/rfc2616.

GOLDING, J., 2014. Blueprint basics. https://www.unrealengine.com/blog/blueprint-basics.

JOHN DEERE WELLAND WORKS. 2006. Compressing the Time Cycle. Tech. rep., MSC Software.

LACOURSE, D. 2003. virtual prototyping pays off. *Cadalyst* (May).

MA, D., ZHEN, X., HU, Y., WU, D., FAN, X., AND ZHU, H. 2011. Collaborative Virtual Assembly Operation Simulation and Its Application. In *Virtual Reality & Augmented Reality in Industry.*, D. Ma, J. Gausemeier, X. Fan, and M. Grafe, Eds. Springer Berlin Heidelberg, 55–82.

MOODBUILDERS, 2015. MoodBuilders. http://www.moodbuilders.com/#!virtual-reality/c20ks.

NVIZAGE, 2006. Nvizage previsualization. http://www.nvizage.com/.

RYAN, R. R. 1999. Digital testing in the context of digital engineering "functional virtual prototyping". *VDI Berichte*, 1489, 163–185.

SCHULZRINNE, H., RAO, A., AND LANPHIER, R., 1998. Real Time Streaming Protocol (RTSP). http://tools.ietf.org/html/rfc2326.

SOWIZRAL, H., ANGUS, I. G., BRYSON, S., HAAS, S., MINE, M. R., AND PAUSCH, R. 1995. Performing work within virtual environments (panel session). In *SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, S. G. Mair and R. Cook, Eds., 497 – 498.

TAKE4D, 2014. Pre-Visualisation / Virtual Production / On-Set Visualisation / Motion Capture / Motion Control Robotics. http://take4d.com/.

THE THIRD FLOOR, 2007. What is previs. http://www.thethirdfloorinc.com/#home-7a225.

WORLDVIZ, 2002. Case Studies. http://architecture-interactive.com/case-studies.

YANG, B. 2015. Practical Virtual Reality for Disney Theme Parks. Game Developers Conference 2015.