Gravity Games - A Framework for Interactive Space Physics on Media Facades

Marcel Köster
xm:lab
Academy of Fine Arts Saar
Saarbrücken, Germany
m.koester@xmlab.org

Michael Schmitz
xm:lab
Academy of Fine Arts Saar
Saarbrücken, Germany
m.schmitz@xmlab.org

Sven Gehring
UMTL
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
sven.gehring@dfki.de

ABSTRACT

We present a framework for setting up interactive three-dimensional outer-space simulations for media facades and other display environments. The core of this framework consists of a space simulator that is capable of efficiently rendering large amounts of objects and computing their physical behavior in real time. The extensible framework further allows to map 3D scenes to multiple displays and integrates interaction components via the TUIO protocol. We present a first use case that uses a mobile-phone GUI as a basis for user interaction on a projection-based 3D media facade. As an initial scenario, we use a generated space scene, which consists of asteroid belts with colliding space objects.

Author Keywords

Space simulation; gravity simulation; media facade; virtual environment; interaction.

ACM Classification Keywords

H.4 Information Systems Applications: Miscellaneous

INTRODUCTION

Over the last decade, an increasing number of digital technologies have been deployed to public spaces. Prominent examples are large-scale digital displays and *media facades* that have been embedded into the urban landscape [5, 7, 16, 23]. These large-scale urban screens usually display both dynamic and static contents as new communication channels to citizens and visitors. Such communication platforms enhance existing architectural structures through digital technologies and novel presentation and sensing possibilities. Digital arts, visualizations, interactive systems or mere announcements are among the diverse types of content. Designers and content providers usually address media facades with planar, 2D form factors. However, interactivity is rarely offered [24]: It poses additional technological and conceptual constraints to the development of contents. As one of the media-facade design challenges, Dalsgaard and Halskov state that media facades need to be integrated into physical structures and surroundings of the underlying building. In accordance, typical media facades tend to cover more than one side of a building, resulting in a non-planar, 3D form factor. We can exploit such a 3D form factor for the display of 3D content on a media facade in general. An example for that is the media facade of the Academy of Fine Arts Saar (Figure 1).

Figure 1. Media facade at the Academy of Fine Arts Saar

In this paper we present a framework to compute and render physical force-of-gravity simulations on media facades. As a use case, we chose a space simulation with many space objects on off-the-shelf graphics hardware. Our framework also supports mapping of the scenery to arbitrary projection-based media facades and other display environments. In order to support interaction, the framework is capable of handling TUIO [17] events — the de facto standard protocol for multi touch input — in order to manipulate objects, processes and parameters.

In the remainder of this paper we will describe the setup of the media facade and the space-simulation framework. We further report and discuss the experiences we gathered by deploying the system to the aforementioned media facade.

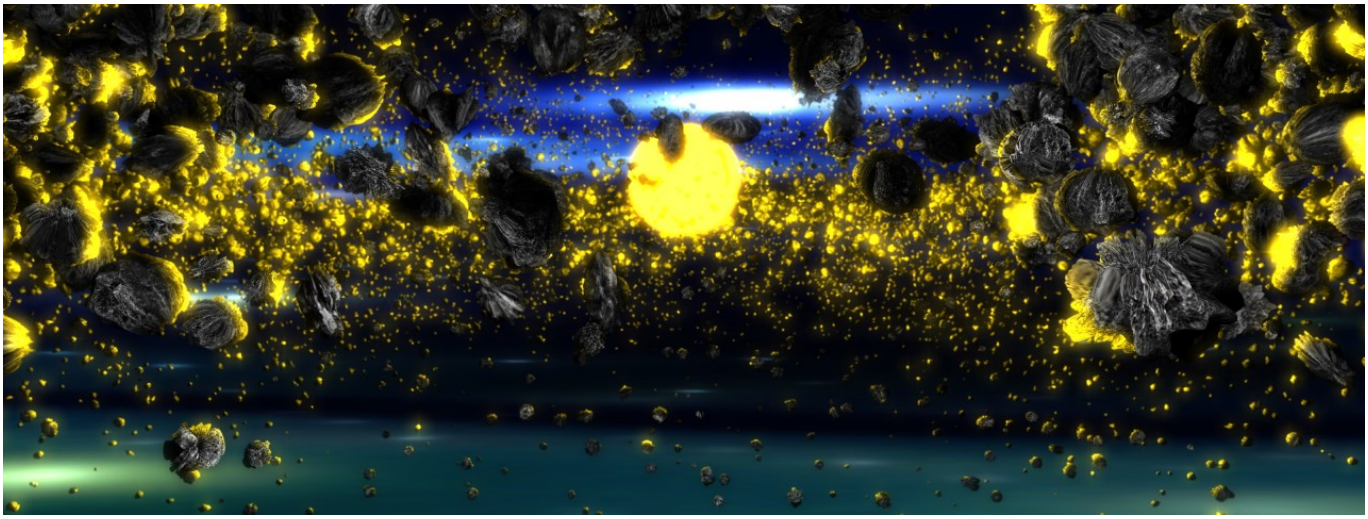

Figure 2. Central part of a rendered image for the target media facade with proper camera setup

RELATED WORK

Media Facades

Fischer et al. investigated spatial aspects in the design of shared encounters for media facades [10]. They introduced the term *Urban HCI*. Furthermore, they provided a formal model which describes spatial configurations for different scenarios, in which the built environment, the interface, any associated computer system and the social context are taken into account [10].

In [4], Boring et al. described a way to use the concept of *Touch Projector* [3] to allow for a simultaneous interaction with a media facade for multiple users through live video on mobile devices. They segmented the overall media facade by defining two rectangular interactive areas (corresponding to the particular sides of the building) which were used as separate, independent interactive areas. These areas were fully within the field of view of users standing in front of them. Introducing *MobiSpray*, Scheible et al. utilized a smart phone as a virtual spray can [22]. They used a world-in-miniature interface in combination with a large-scale projected media facade to spray virtual colors on various surfaces. Baur et al. [2] used smart phones to apply the metaphor of optical projection to post visual content onto various digital surfaces in public multi-display environments. With *spread.gun* [11], Fischer et al. built a stylized stationary cannon for shooting color dabs onto a projected media facade. In [12], Fischer et al. adapted this approach by exchanging the stationary input device for a mobile slingshot to support social interaction between users, since they had to pass around the input device. However, the aforementioned approaches for interacting with a media facade are usually limited to planar 2D form factors.

Dalsgaard and Halskov identified eight challenges for designing urban media facades [9]. They state that media facades need to be integrated into physical structures and surroundings. This often leads to media facades that cover more than one side of a building, resulting in a 3D form factor. To exploit the full potential and capabilities of these facades, the ultimate goal is to offer the opportunity for a fluent and continuous interaction with the whole facade. This includes interaction over borders and around corners. With [14], Gehring and Krüger provided an approach to enable users to interact with the complete media facade — including its occluded parts — with the help of a mobile device. They used cartographic map projections to create an interactive 2D map representation of the media facade's 3D surface.

Space Simulators

Space simulators have been developed to allow for an interactive exploration of gathered data sets and the simulation of fictional and real scenarios in space. There are many applications that offer both, like the freely available *Space Engine*, which targets desktop computers [21]. This application can simulate different galaxies, planets and moons, for instance, including their motion in real time. An observer has the possibility to interactively explore the visualized space in a first-person perspective using the default computer input methods (mouse and keyboard). Known galaxies are loaded from datasets which include planetary surface information and textures. Unexplored parts of the universe are created procedurally. Lighting effects, volumetric nebula and the visualization of orbital paths enhance the immersion. However, this engine focuses on desktop computers and on the exploration of objects in space, not on experiments with gravity or sophisticated multi-display setups for media facades.

Another interesting interactive application is provided by the NASA in form of *Cassie 3D* [18]. It is freely available and is tailored for the exploration of the Saturn Cassini mission. However, it only focuses on this particular mission without providing the possibility to simulate other space scenarios or the interactive exploration of the deep space.

The application that is most similar to our approach is the *Universe Sandbox*, which offers a real-time gravity simulator [15]. The physics engine simulates the orbital motion of objects in space based on Newton's laws of gravitation. By experimenting with the simulation, players can explore

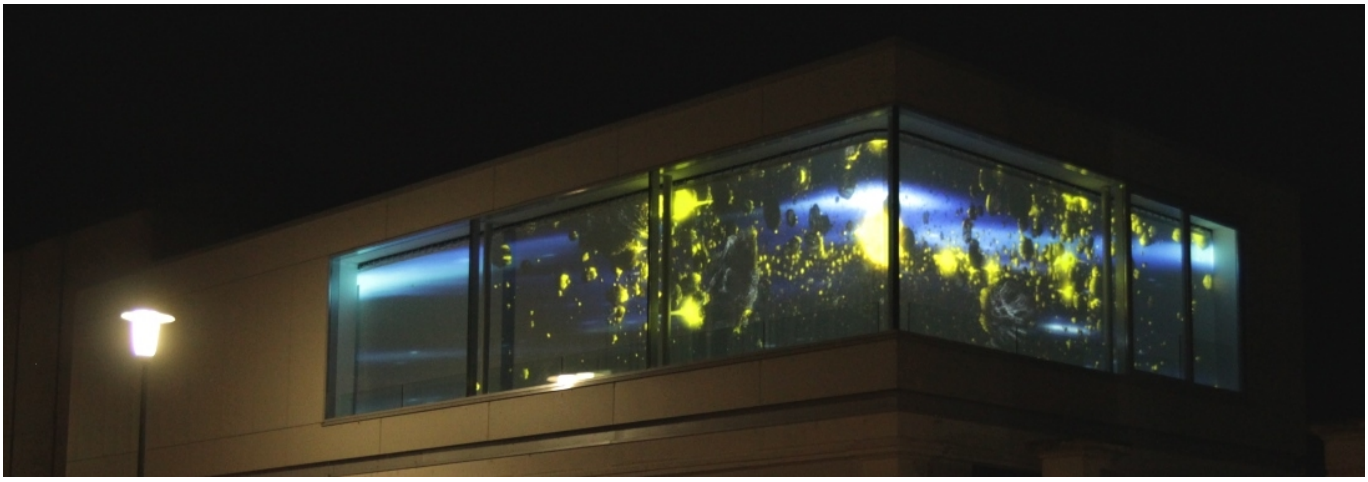

Figure 3. Image of the simulation on the target media facade

the universe and play with the stars: Complex collisions and chain reactions between objects are possible. In addition, a player can manipulate gravity or spawn new asteroids during runtime through desktop-interaction events. Furthermore, the focus of this application lies on the simulation features and not on visually appealing rendering of objects in space.

3D SPACE-SIMULATION FRAMEWORK

Each media facade requires a unique adaption in order to fit the prerequisites of the facade, as well as the requirements of the underlying hardware realization. This includes specific virtual-camera settings, display setups, multi-display support, proper perspective distortions and support for the available interaction capabilities.

Our space-simulation framework is realized in the form of a lightweight extensible engine, which is designed to fulfill these requirements to target media facades and other virtual environments. In order to ensure extensibility, it offers a plugin system to extend, adapt and customize several parts of the simulation engine. This includes the realization of custom scenarios (pure simulations or interactive games, for instance), custom interaction logic or specific rendering features. For this reason, the whole framework is split into two major parts: the application itself and an engine-interface library, which is referenced by plugins. The application contains the core physics, graphics and interaction-processing functionality, whereas general interfaces and settings are shared between the application and the library.

Rendering

The simulation has to render many objects but should still be able to visualize them in an appealing way for an observer. Furthermore, there are different types of objects in space, which should appear as differently looking objects as well. Since many scenarios require multiple virtual cameras to render the simulation from different perspectives in parallel, most or nearly all objects are visible with respect to the camera setup. Also, in contrast to most common interactive 3D applications that typically render only a small excerpt of the world, we visualize a much higher percentage of the whole scene on a facade. Therefore, we assume that all objects are visible in parallel.

In order to satisfy these requirements, we designed and integrated a dynamic level-of-detail based rendering approach for our scenario. It leverages procedurally generated geometry, which allows for an appealing visualization of planets or asteroids. However, in many scenarios it is also often required to support objects with a predefined geometry, which is given in the form a of discrete mesh. For instance, in cases in which the geometry cannot be described easily for a non-expert in a procedural way. Our approach, to render those objects appealingly, involves grouping of objects into LOD groups, according to their distance to the camera, which turned out to be much faster in our case than other more sophisticated hierarchical-based approaches [8]. We perform a grouping step on the GPU while taking the LOD information of every virtual-camera perspective into account and perform the rendering step afterwards using geometry instancing [20].

Mapping to output devices

As the simulation should be adaptable to different facades, multiple render outputs (for example, connected projectors) with arbitrary layout and distortion of the output surfaces, have to be supported. An additional requirement is the use of multiple virtual-camera perspectives in parallel. This is particularly useful when targeting facades with multiple projection surfaces that are not contiguous. This in turn means, that we have to render as many images as there are virtual cameras. An example for such a scenario can even be a 360-degree rendering requiring six distinct camera perspectives in order to build a complete 360-degree image. Furthermore, round form factors are also supported when adjusting the required distortion and camera setup.

In order to send the proper contents to the different connected output devices, we have to adjust the different images properly on the virtual-desktop workspace locations. The underlying operating system in combination with the graphics driver will automatically send the contents to the proper output de-

vices. Note that it is also possible to span parts of the scene across multiple output devices.

An alternative to this approach is merging or combining the connected output devices to a large single virtual desktop. However, this often causes software incompatibility issues with other programs (like media players) since we have to override the default window-mapping functionality. Moreover, this does not solve the mapping problem completely: We still have to render different camera perspectives and arrange them in a proper way. In addition, we want to be less invasive and do not want to manipulate the virtual-device setup of a particular media facade since other simulations or videos are already adapted to a particular setup.

Post-process image-distortion can be applied to specific regions of the final image. This helps to support arbitrary display form factors. An example for such a process can be a rectangular image that has to be distorted in a specific way to match the target layout of a projected image.

The distortion process itself is not limited to a mathematical operation (like shearing) or a set of cascaded operations. Instead, a distortion mask can be defined or imported based on images or 3D triangular meshes. However, the required mask has to be adapted for every target media facade or interactive installation currently.

Physics

The physics engine implements a force-of-gravity simulation in space. The gravitational-force computation between two objects is given by Newton's law of gravitation

$$\vec{f}_{21} := G\frac{m_1 m_2}{|\vec{d}_{12}|^2}\hat{d}_{12} = G\frac{m_1 m_2}{\vec{d}_{12}\cdot\vec{d}_{12}}\hat{d}_{12},$$

where

- $\vec{f}_{21}$ refers to the force object 2 applies on object 1,
- G is the gravity constant $\approx 6.674 \cdot 10^{-11}\frac{m^3}{kg\cdot s^2}$,
- m_1 and m_2 are the masses of object 1 and 2,
- $\vec{d}_{12}$ is the distance vector between the two objects and
- $\hat{d}_{12}$ is the normalized distance vector.

This kind of simulation requires a computation of the gravitational forces between n different objects (bodies): Every body influences the force that is applied to the other bodies. We can use the massively parallel nature of the GPU to compute those forces in parallel, and thus, in real time. The actual computation is performed by an algorithm similar to the one by Josh Barnes and Piet Hut [1] while using arithmetic, data-structure and data-access pattern specific optimizations to improve the performance on GPU accelerators (see also [19]). This ensures an efficient computation on current GPUs.

Interaction

The interaction part of the framework currently supports three general input methods: The standard protocol for tangible user interfaces (TUIO) [17], motion tracking and default desktop interaction methods. The modules of the framework that retrieve interaction events can be run on other machines than the main simulation computer.

We decided on the integration of TUIO since there are many client applications and devices that support this protocol. Besides other features, TUIO offers the opportunity to interchange information about touch and drag events. For example, a touch event can cause an object to be selected in the client application whereas a drag event can move an object.

Motion tracking supports default optical, infrared and depth cameras. We support default and infrared cameras via the USB protocol, the Microsoft Kinect [25] via the native interfaces provided by the Microsoft Kinect SDK and other depth cameras via the Open Natural Interaction (OpenNI) [1] framework. The processing and the extraction of motion information of the received image data is realized using the Open Source Computer Vision Library (OpenCV) [6].

Implementation

The whole framework is implemented in C++. Computations on the CPU make use of SSE and AVX instructions to improve performance. Furthermore, we leverage Direct3D 11 [26] for the rendering part in order to benefit from tessellation capabilities. We use Direct Compute [26] for computations on the GPU, which integrates seamlessly into the Direct3D functionality. It is similar to other well known APIs/programming languages which are available for this task like OpenCL [2] or Cuda [3].

Figure 4. The media facade's 3D effect running a demo video

USE CASE: MEDIA FACADE

The first use case of our simulation framework is the media facade of the Academy of Fine Arts Saar. The facade is about 3.4 meters high and 20 meters wide. It is located about 5.5 meters above the ground and spans over five windows of equal size, as shown in Figure 1. Multiple projectors project media contents from the back onto a canvas behind the windows. The canvas is mobile and driven by an electric

[1] `http://structure.io/openni`, Accessed April 2015
[2] `http://www.khronos.org/opencl`, Accessed April 2015
[3] `https://developer.nvidia.com`, Accessed April 2015

motor such that it can be moved aside during the day automatically. The schedule for the projections and the control of motor is managed automatically via a software interface. A mobile back projection curtain offers the opportunity to use the space inside the building as a traditional gallery for exhibitions and presentations at daytime.

The whole canvas is subdivided into five smaller ones, where every small canvas covers a single window. In our setup, a single projector is used for every window, which sums up to a total number of five projectors for the whole media facade. The projectors are mounted to the ceiling and are oriented in a way that every projector is perpendicular to the projection surface.

When looking at the enclosed edge, a spectator gets the impression of volumetrically spreading 3D objects, which seem to be caught within the room confined by the projection canvases (Figure 4). Note that this 3D illusion is also perceptible without the use of additional hardware like active or passive 3D glasses. This setting is perfectly suitable for a wide range of different observers which briefly stay and watch. In order to support this effect, a proper virtual-camera setup that takes this perspective into account, is required. The ideal observation point is located on the opposite side of the road, as shown in Figure 5. At the same time, this is a limitation to this 3D approach, since it relies on fixed observer-camera angles for a proper perception of the 3D illusion.

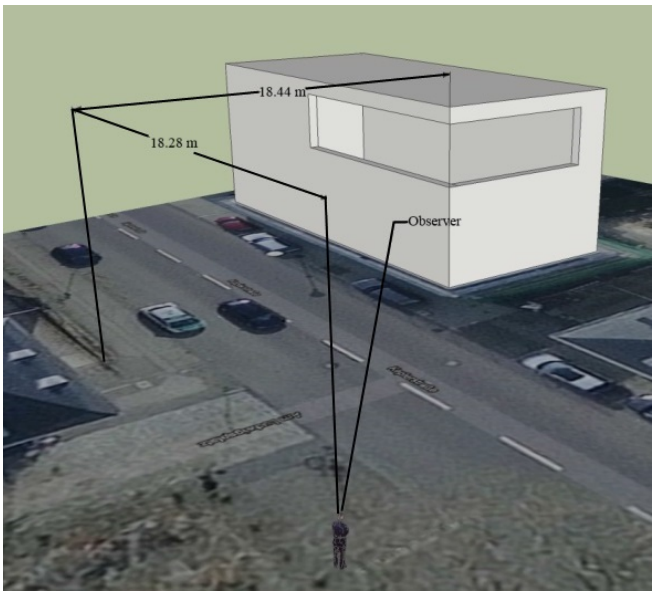

Figure 5. Observer position to perceive the simulated 3D effect

Technical Setup

As mentioned previously, the projection canvas consists of five projectors. Each projector is typically used with a resolution of 1024×768 pixels, which can be increased to 1400×1050. Those projectors are connected to a default off-the-shelf middle-end graphics accelerator from AMD which can handle up to six output devices in parallel. The sixth output device is used for maintenance and/or development services.

Projection Mapping

In order to reflect the physical form factor of the facade, the images have to be distorted according to a specific mask. A sketch of the used mask is shown in Figure 6. This defined mask, which is given by the projection environment, is used for extraction of regions from the rendered images.

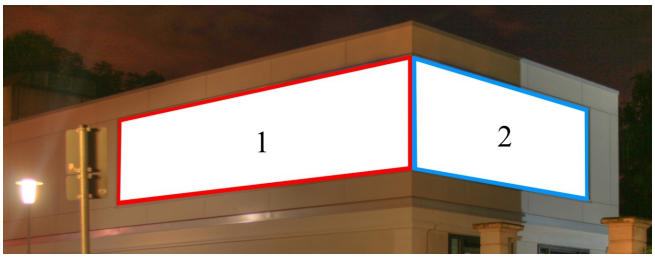

Figure 6. Sketch of the used mask for projection mapping

In our scenario, we need to extract two regions of the final image (one region for each side of the facade), which are shown in Figure 7. The black border indicates the rendered image. After the extraction, we can distort those regions to rectangles in order to send them to the output devices (Figure 8). The actual projection process performs the inverse transform: it projects the rectangles to the projection screens that appear to be distorted from an observer's point of view. Using this transformation, an observer can exactly see what we wanted him to see: a properly mapped image with the correct perspective adjustments.

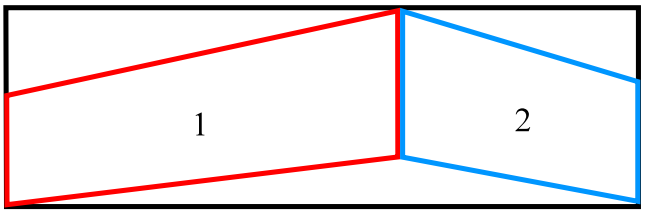

Figure 7. Extracted regions according to the used mask

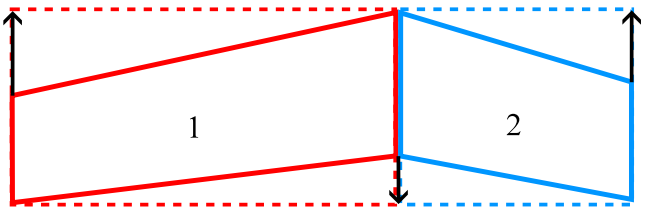

Figure 8. Distortion/deformation according to the used mask

Simulation Scenario

We designed a scenario which consists of an asteroid belt with an initial number of $2^{14} = 16384$ asteroids. It orbits around a central sun. We use different object types via displacement mapping to simulate different asteroid geometries and different surfaces (based on material properties and textures). Already a small number of nine predefined types provides the impression of many different asteroids. In this case we used pre-designed materials and textures[1] for the asteroids

instead of procedurally generated ones. The object geometry is based on different procedurally-generated ICO spheres (icosahedrons) which allow for an appealing displacement of asteroids during rendering.

The mass of each asteroid is responsible for its size, which makes it easy for an observer to distinguish between heavy (large) and light (small) objects. Furthermore, the mass controls the collision behavior, which is based on the laws of physics and every collision causes additional visual effects to appear. Based on a comparison of the mass relation, some objects might be smashed whereas others remain while smashing their collision partners. Objects that have to be smashed will be subdivided into smaller ones which are added to the simulation again. However, the subdivision logic will only be applied if the asteroid is large enough to avoid endless splitting. A collision with a sun always results in a complete destruction of the asteroid. An average number of objects in space during the simulation (without an interaction event) is around $45500 \approx 2^{15.47}$.

The interaction component of the simulation allows to interfere with this scenario. We allow a manipulation of a sun and the creation of additional asteroids on a mobile device via the TUIO protocol. The manipulation capabilities allow to experience major changes in the gravity field while destabilizing the asteroid belt. Every mobile-phone user that connects will automatically create a new sun with a unique color whereas the first user controls the central sun. In the case of a longer period of time without external interaction events (target audience are passers-by), the simulation logic creates random asteroids and tries to re-stabilize the belt.

In order to benefit from the form factor of the facade, we adapted the virtual-camera setup and the final image distortions. The center of the simulated system should be in the center of the visualization, as the belt rotates around a central sun. If the mapping of the sun is performed naively, the projected virtual position will be located on the window surface, as shown in Figure 9. This results in a viewing angle between the virtual camera and the observer of around 15 degrees.

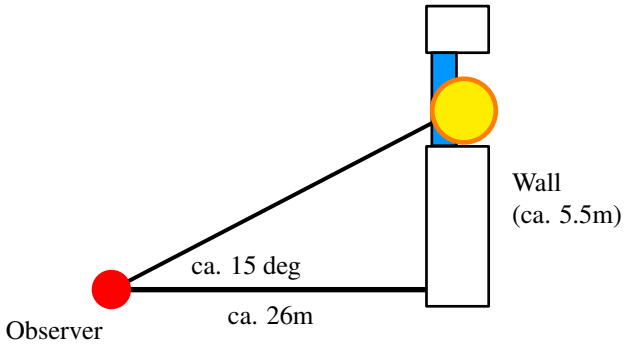

Figure 9. Naively projected position of the simulated sun

Unfortunately, this camera setup does not allow for the desired 3D effect, in which the sun seems to be centered in the virtual room. In order to adapt the setup, we need to take the front edge of the facade into account (see also Figure 5). The virtual-camera position and its angle have to be adjusted in order to match an observer's viewing perspective in reality (Figure 10). An observer (who is approximated by the height of the observer's eyes of about 1.6 meters) looks at the sun through the windows at the edge of the facade. This results in an angle between the observer and the center of the windows of about 8 degrees.

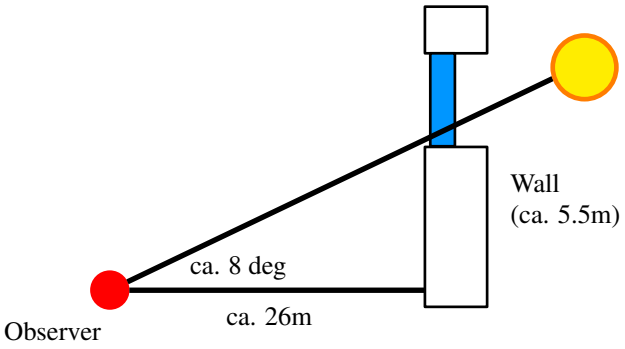

Figure 10. Sketch of the observer's viewing perspective who looks at the virtual position of the simulated sun

A rendered image for the media facade (including proper camera setup) before the final projection-mapping-distortion step, is shown in Figure 2. A picture of the deployed simulation on the media facade is shown in Figure 3.

CONCLUSION

In this paper we presented a framework for simulating large-scale 3D content on media facades with non-planar form factors. Our framework provides customization and adaption capabilities to address the variety of different media facades and environments. The integration of interaction features through the generic TUIO protocol was tested with mobile devices as well as motion tracking input, controlling or creating space objects in real time. Supporting the TUIO protocol turned out to be a valuable choice, as it reduces the complexity of integrating different interaction clients. The process of mapping virtual cameras to real projection surfaces and adjusting image-distortion masks (if required) is still complex and difficult to automatize.

For further testing and simulation, the output of the system can be connected to the *Media Facade Toolkit* of Gehring et al. [13] to inspect whether the 3D effect works for the intended position of the observers. This is a particularly crucial issue for media facades that are optimized for certain viewing positions, directions, or distances.

For the future, we will investigate alternative interaction techniques, such as shifting the virtual-observer position. Furthermore, we would like to test the system on media facades of different form factors and other interactive installations. We also plan to create new scenarios like a recreation of our solar system and offer interaction possibilities that allow for playful learning sessions. Users would then be able to destroy planets or to change the size of the sun and see the effects on the orbits of the other planets and moons.

[1]These materials were made by Pascal Klein, a designer from the Academy of Fine Arts Saar.

REFERENCES

1. Barnes, J., and Hut, P. A hierarchical O(N log N) force-calculation algorithm. *Nature 324* (1986).
2. Baur, D., Boring, S. and Feiner, S. Virtual Projection: exploring optical projection as a metaphor for multi-device interaction. In *Proceedings of CHI'12*, ACM (2012).
3. Boring, S., Baur, D., Butz, A., Gustafson, S. and Baudisch, P. Touch Projector: Mobile Interaction through Video. In *Proceedings of CHI'10*, ACM (2010).
4. Boring, S., Gehring, S., Wiethoff, A., Blöckner, M., Schöning, J. and Butz, A. Multi-User Interaction on Media Facades through Live Video on Mobile Devices. In *Proceedings of CHI'11*, ACM (2011).
5. Bouchard, D. *Embodied Emergence: Distributed Computing Manipulatives*. Sciences, 2007.
6. Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).
7. Bullivant, L. *Responsive Environments: Architecture, Art and Design*. V&A, London, 2006.
8. Clark, J. H. Hierarchical geometric models for visible surface algorithms. *Commun. ACM* (1976).
9. Dalsgaard P. and Halskov, K. Designing Urban Media Facades: Cases and Challenges. In *Proceedings of CHI'10*, ACM (2010).
10. Fischer, P.T. and Hornecker, E. Urban HCI: Spatial Aspects in the Design of Shared Encounters for Media Façcades. In *Proceedings of CHI'12*, ACM (2012).
11. Fischer, P.T., Zöllner, C. and Hornecker, E. VR/Urban: Spread.gun - design process and challenges in developing a shared encounter for media facades. In *Proceedings of British HCI'10*, ACM (2010).
12. Fischer, P.T., Zöllner, C., Hoffmann, T. and Piatza, S. VR/Urban: SMSlingshot. In *Proceedings of TEI'10*, ACM (2010).
13. Gehring, S., Hartz, E., Löchtefeld, M., and Krüger, A. The media facade toolkit: Prototyping and simulating interaction with media facades. In *Proceedings of UbiComp'13* (2013).
14. Gehring, S., and Krüger, A. Facade map: Continuous interaction with media façades using cartographic map projections. In *Proceedings of UbiComp '12* (2012).
15. Giant Army. *Universe Sandbox*. http://universesandbox.com/, Accessed April 2015.
16. Haeusler, M.H. *Media Facades - History, Technology, Content*. avedition, 2009.
17. Kaltenbrunner, M., Bovermann, T., Bencina, R., and Costanza, E. *TUIO: A Protocol for Table-Top Tangible User Interfaces*. In *Proceedings of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation* (2005).
18. NASA. *Cassini*. http://saturn.jpl.nasa.gov/multimedia/CASSIE/, Accessed April 2015.
19. Nguyen, H. *GPU Gems 3*. Addison-Wesley Professional, 2007.
20. Pharr, M., and Fernando, R. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, 2005.
21. Romanyuk, V. *Space Eengine*. http://en.spaceengine.org/, Accessed April 2015.
22. Scheible, J. and Ojala, T. MobiSpray: mobile phone as virtual spray can for painting BIG anytime anywhere on anything. In *Proceedings of SIGGRAPH'09*, ACM (2009).
23. Schoch, O. My building is my display. In *Proceedings eCAADe'06* (2006).
24. Struppek M. Urban Screens - The Urbane Potential of Public Screens for Interaction. In *"Screenarcadia" - Microwave - International New Media Arts Festival, festival book* (Hong Kong, 2010).
25. Zhang, Z. Microsoft kinect sensor and its effect. *IEEE MultiMedia* (2012).
26. Zink, J., Pettineo, M., and Hoxley, J. *Practical Rendering and Computation With Direct3D 11*. CRC Press, 2011.