3D DATA TRANSFER GAME JAM

Report on an international interdisciplinary workshop on the transfer of Rhino 3D files into games.

ABSTRACT

Modern game engines such as Unity or Unreal enable rapid 3D prototyping and are currently changing the way analogue products are designed and perceived. Game Engines, however, require a higher level of visual accuracy than is typically provided by packages commonly used in the 3D visualisation industry such as Rhino 3D. In this paper we report on an interdisciplinary international student exchange project which took place in the winter semester 2021/22. The goal of the exchange was to foster interdisciplinary projects and skill transfer among design and computer science and media students. The goal was to use existing Rhino 3D project files as source data for game productions. The results, five games and detailed discussions of opportunities and challenges phased during data transfer from Rhino 3D to gamed engines, are outlined in this paper.

AUTHORS:

SABIHA GHELLAL PH.D. (HOCHSCHULE DER MEDIEN) – PROFESSOR FOR EXPERIENCE & GAME DESIGN

CLAUDE SAOS (LISAA: L'INSTITUT SUPÉRIEUR DES ARTS APPLIQUÉS) - HEAD OF INTERIOR ARCHITECTURE AND DESIGN DEPARTMENT

STUDENTS NAMES ARE ASSIGNED TO EACH PROJECT

PORJECT VIDEO REPORT:
https://www.hdm-stuttgart.de/mediathek/projectpage/3847/details

Introduction
Modern game engines such as Unity or Unreal enable rapid 3D prototyping and are currently changing the way analogue products are designed and perceived. The use of game engines by designers and engineers began with the shift from analogue to digital prototypes. Whether in consumer products, automotive, or architecture, engineers are increasingly relying on 3D models instead of physical models for product development and design decisions. Game Engines, however, require a higher level of visual accuracy than is typically provided by packages commonly used in the 3D visualization industry such as Rhino 3D[1]. Editing in game engines, usually based on polygon and mesh-based models, is not a required skill for designers using parametric CAD software. In the past, the steep learning curve kept many CAD users from venturing into Unity or Unreal Engine, the current two leading game engines. As game engine manufacturers have moved into new markets, they have begun to incorporate additional features for specific applications and industries, making product or even production design with game engines accessible. Today, game engine features such as "Unity Forma[2]" or "Unreal HMI[3]" allow engineers and designers to take advantage of rich game development tools without having to learn the code and lingo of game development.

In order to facilitate the changing requirements in 3D visualization it would be possible to simply change tools, and start teaching Unity or Unreal for 3D visualizations instead of Rhino 3D to students and practitioner accordingly. However, without looking into the specific industry requirements and specific features of the tools currently used in the industry this may be a premature decision. Therefore, we organized an international and interdisciplinary project dealing with data transfer and management between different tools and invited 18 students with skills in 3D design and game engine development.

Game Jam and Projects
In order to facilitate and enable praxis-based education and provide essential future perspective for the education of our students, we organised a series of workshops and "Data Transfer Game Jam". In this international and interdisciplinary project, the one-week "Data Transfer Game Jam" took place as part of a Master's course in Computer Science and Media at the Hochschule der Medien (HdM) together with the art academy LISAA (L'Institut Supérieur des Arts Appliqués). In preparation for the Game Jam, there were three keynote presentations and workshops on game design, 3D design and game development using game engines. The goal of the "Data Transfer Game Jam" was to design and implement 5 Rhinoceros 3D CAD projects as games. We left it up to the interdisciplinary teams to decide what games they want to develop. The task of the computer science and media master students was however not only to participate in the design and implementation of the games, but also to pay special attention to the data transfer and data management. The curses selected for this interdisciplinary project included bachelor program lecture Interior Architecture & Design from LISAA and a master program lecture game design/development from Hochschule der Medien. This paper describes the five projects and the necessary data management and data transfer.

Curriculum Integration
The focus of the interdisciplinary praxis-based lectures was the understanding, integration, deepening and development of 3D games. This included the analysis of existing 3D data and matching games, conceptualization of game design ideas based on the previously completed analysis, realization of game design concepts and development with game engines. On a more practical level the focus of the workshop and game jam was to get to know the tools used in the two different industries and work together to transfer data from Rhino 3D to commercially available and for educational purposes free of use game engines.

To facilitate interdisciplinary exchange, we asked students to work with existing designs developed by LISAA students in a previous semester, rather than producing all content collaboratively. The goal was to allow students to focus on data transfer and preparation rather than content production, with, for example, game art for game development. In addition, HdM master's students were assigned to teach LISAA students basic game engine programming skills and to investigate and document the various data transfer requirements described in this paper.

Realised Projects
Five projects will be presented by the involved students. "Escapers" a first-person puzzle, "Palma" a three-dimensional real-time mobile strategy game, "Starsphere" a mobile tilting game, "Save the Bees" a tower defence game and "Desert" a first-person exploration game with puzzle elements.

[1] https://www.rhino3d.com
[2] https://unity.com/de/products/unity-forma
[3] https://www.unrealengine.com/en-US/hmi

Escapers

Manuel Fankhänle (HdM),Christof Schwarzenberger (HdM), Timothèe Guyot (LISAA), Yassine Fellah (LISAA), Recep Dogan (LISAA)

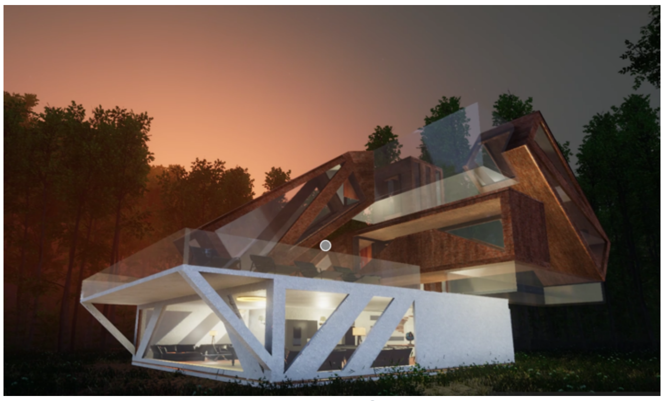


Figure 1 First Person Puzzle Exit Game

Escapers is a first-person puzzle PC exit game (Figure 4) that has since been exported as a virtual reality game and was exhibited at the Game Zone of the 29th International Festival of Animated Film in Stuttgart, Germany. The challenges associated with a first-person real-time 3D game differed from some of the other projects developed under this initiative. Since the player was required to enter the 3D building in order to find an exit. Therefore, LISAA students had to provide additional models for the game, including furniture and other interior design elements. The original architectural idea of the 3D design exercise was to create a utopian building, a kind of brutalist architectural maze, see figure 5.

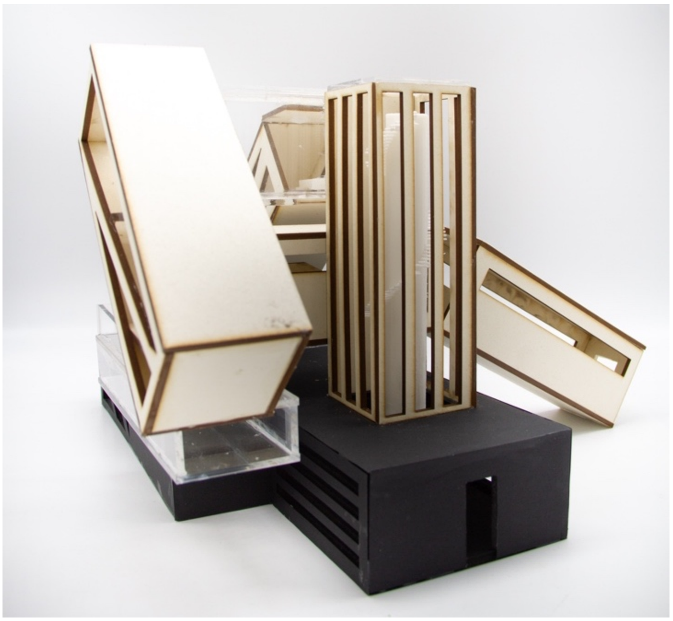


Figure 2 The original architectural Concept

Rhino 3D can be used to create, edit, render and animate polygon meshes. This allows to generate detailed models with a high complexity. However, transferring data for the purpose of real-time use in a game presented some challenges, which we will explain below.

Escapers Data Transfer

As a first step we transferred all 3D Data into Blender[4] files. Blender is a free and open source 3D application used for modelling, rigging, rendering, animation, simulation, motion tracking and allows for low poly options often necessary for game and other real time content creation. The Blender files where then imported in the game engine Unity[5]. Unity is a development environment which can be used to create and program interactive and three-dimensional games for multiple platforms. Figure 6 illustrates the data transfer necessary to create a 3D first person interactive experience in a game engine.

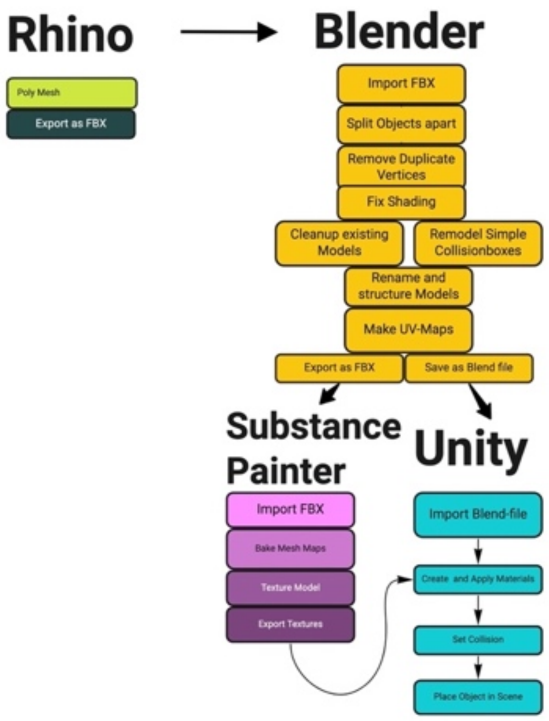


Figure 3. Data Transfer

Firsts of all, it was noticeable that translations and transformations of faces, edges and vertices manipulated the 3D meshes in an unexpected way. Duplicate vertices and disjointed faces became visible. In a demonstration in Rhino 3D, the designer showed how models were created. Here the designers did not use a single mesh to transform the target shape. They used multiple meshes that added to the original mesh or cut recesses in it. This resulted in multiple vertices and faces that did not belong together due to the unconnected meshes used in the modelling process. The first step in

[4] https://www.blender.org

[5] https://unity.com/de

solving this challenge was to split the individual rooms and non-contiguous models into separate models. To fix the issue with the duplicate vertices, the ones that are in the same position were merged. However, this resulted in problems with the shading of the model. In Blender, the "Auto Smooth" feature takes care of fixing the shading artefact.

Another challenge was the high complexity of the models. The designers used the “Split Edge” feature in Rhino 3D to smooth edges of the meshes which led to a high number of vertices and polygons. This is unfavourable in game development because the models have to be rendered in real-time. Since the individual vertices have to be calculated later, more complex models lead to a higher computational effort. To reduce complexity, close vertices were merged and shading issues were fixed with “Auto Smooth”, as in the shading problem. Blender provides smoothing features that do not actually modify the object’s geometry. It changes the way the shading is calculated across the surfaces (normals will be interpolated), giving the illusion of a smooth surface [6].

The model still had many vertices that did not contribute to the structure. To remove these, the “Remesh” and “Retopology” tools were used in Blender. Remeshing is a technique that automatically rebuilds the geometry with a more uniform topology[7]. Retopology is the process of simplifying the topology of a mesh to make it cleaner and easier to work with. Through the designers' demonstration of Rhino 3D, it is clear that there is less focus on geometric layouts or accuracy. The tool was used more for visualizations that focus on design intent.

The last problem was some artistic liberties taken by the designers. The abstract design of the building had to be explorable by the player later and work in the game. For this, all rooms had to be made accessible, game areas had to be limited (e.g. stairs leading into the void) and doors had to be inserted. To prevent the player from falling through floors or running through walls in the game, collision boxes were needed. Therefore, all floors, roofs and walls had to be remodelled with simple box shapes. The collisions of the complex mesh could not have been calculated in real time. All boxes were stored in a separate collection.

At this point the clean-up process of the model in Blender was completed. The clean version of the house was then “UV-Unwrapped”. This process is necessary to assign textures to the faces of the mesh later[8]. There were three different UV-maps. One for the transparent glass, one for the wood parts and the last one for the concrete parts. The final step in Blender was to export the fbx and save the file for Unity.

The fbx was then imported into Substance Painter. There, the model was baked and textured. The textures were saved in 4K resolution. In Unity, the blend file was imported, as later changes could be applied directly from Blender. The textures created in Substance were used to give the materials the right look. The final step was to apply collisions to the collision boxes and place the house in the scene.

Palma

Niels Keller (HdM), Kevin Waldenmaier (HdM)
Agathe Mathis (LISAA), Sarah Sissani (LISAA)

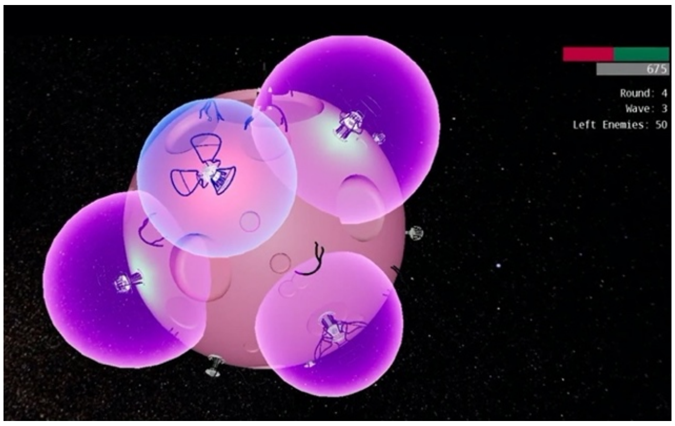


Figure 4 – An abstract architecture turned into a three-dimensional real-time strategy game

In Palma (figure 1) a three-dimensional real-time strategy game for mobile devices, players have to defend their base against robot opponents. To win the game the player needs to build towers of different types. Each tower will help defeating the enemies in different ways, including bullets, slowing down the robots or applying Areas of Effect (AoE) to damage enemies. By defeating an enemy, the player earns points which can be spent for upgrading towers or to build new ones.

Palma Data Transfer

The mobile tower defence game Palma is based on the architectural exercise designed as part of the lecture 3d infography at LISAA (L'Institut Supérieur

[6] https://docs.blender.org/manual/en/latest/scene_layout/object/editing/shading.html
[7] https://docs.blender.org/manual/en/latest/modeling/meshes/retopology.html

[8] https://docs.blender.org/manual/en/2.79/editors/uv_image/uv/editing/unwrapping/introduction.html#about-uvs

des Arts Appliqués). The idea of the abstract building is based on a utopic architecture resembling a flower with its petals and pistil. The petals function as energy collectors to power the central sphere which is the housing unit., see figure 2.

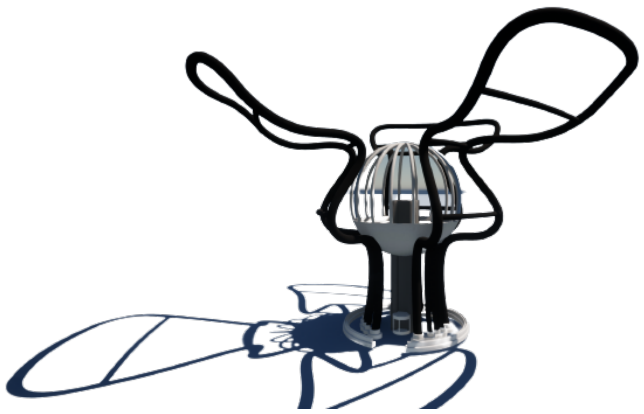


Figure 5 Palma 3D Model in Rhino 3D

The game was programmed with the Unreal Engine 4.27. The main data transfer challenge in this project was the maximum number of polygons to be displayed on mobile devices. Since the tower is used several times in a tower defence game, the number of polygons plays a much greater role and has to be optimized to ensure a smooth performance. In order to simplify the placement of objects in the scene, as well as possible animations, the correct positioning of the pivot point is essential. In our case, the optimal point for the towers was in the middle of the base of the model. For the robot opponents, on the other hand, the optimal pivot point is in the centre of mass, as this is the only way to ensure smooth rotation of the base sphere. Figure 3 illustrates the Data Transfer.

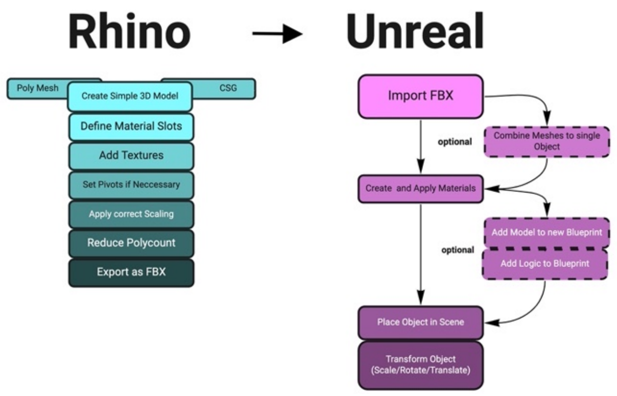


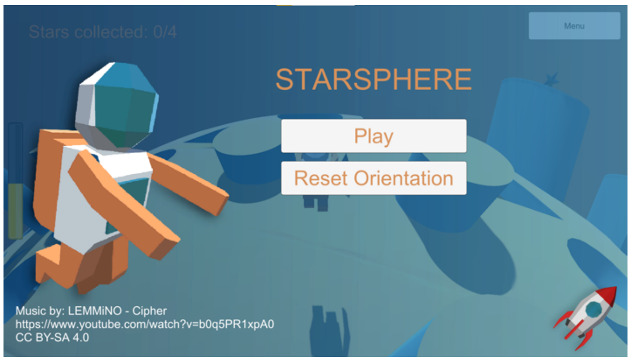
Figure 6 - Palma Data Management

The objects were exported as FBX format from Rhino to allow easy integration into Unreal. In Unreal, the individual components of the objects were combined into a static mesh. This method has the disadvantage that the material slots are still assigned to the individual components and thus redundant materials are applied to the object. This increases the number of draw calls, which is detrimental to performance, especially on mobile devices. Since the required performance margin was available, this solution was chosen despite its challenges due to the simpler workflow.

Various modelling paradigms were used in the project. These include "Constructive Solid Geometry" (CSG) and "Polygon-Mesh". The latter is to be preferred for real-time applications. Since the final object can consist of several individual components with closed geometry in models created using the CSG method, there are unnecessarily many invisible surfaces, which increase the number of polygons and thus harm performance.

Starsphere

Sven Kirsch (HdM)
Léa Hatil (LISAA)



Music by: LEMMiNO - Cipher
https://www.youtube.com/watch?v=b0q5PR1xpA0
CC BY-SA 4.0

Figure 7 Starsphere a mobile platformer

With Starsphere the goal was to create a game with a spherical world as its base. The player should be able to walk on the sphere in every direction and must collect stars to win the game. The Player has to be careful not to fall into “space-holes” which are scattered all over the sphere. To achieve this, the player is able to use a jetpack to jump over small holes or onto elevations. But the jetpack has limited power so the player is forced to find a place to rest and recharge.

The goal of our project was to design a mobile game that uses the orientation (tilting) of the smartphone as an input modality. The number of polygons for a mobile device and how they are displayed differ from desktop applications. All object where created in Rhino 3D with the purpose of creating a utopian architecture using the codes of the moucharabieh that can be found in the architecture of the Arab world, see figure 8. The original purpose was never to interact with the 3D objects using a mobile device. As such the need for data transfer for this project was driven by the special requirements for mobile games.

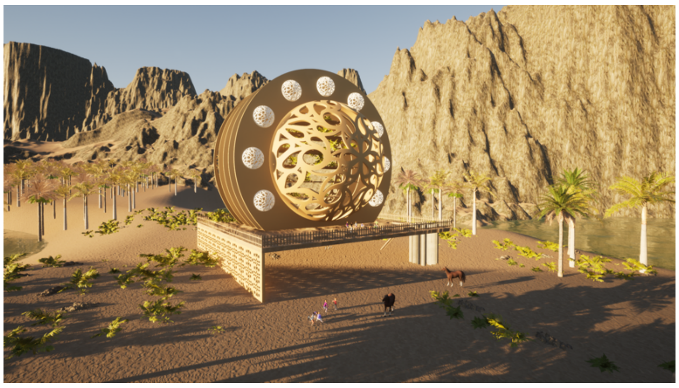


Figure 8 The Rhino 3D Model

Starsphere Data Transfer

To be able to use the objects from Rhino in another programs, they had to be exported to a more universal format. Fortunately, Rhino supports the fbx format, a format supported by most game development tools. The only problem was that Rhino does not necessarily create polygons like other modelling programs while exporting. For example, an absolute flat and rectangular surface could be represented by only 2 triangles.

Rhino, on the other hand, sometimes splits those to fix this problem, we decided to first import the fbx file into Blender and merge all the divided faces and recombine the objects. Also, Rhino may not recognize the normals of the faces correctly and they need to be mirrored.

Figure 9 illustrates the Data Transfer for the Starsphere project.

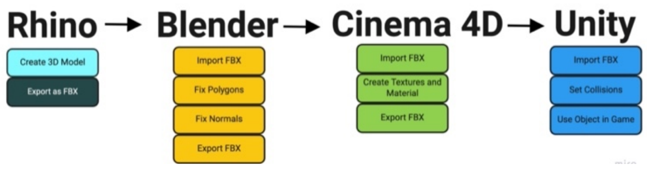


Figure 9 Strashere Data Transfer

After creating the object in Rhino and fixing it in Blender the object could be imported into Unity. Because the object didn't have any textures at this time, it was imported into Cinema 4D first to create the textures. For the game it was decided early on that it should be a low poly style. After the models have been created, cleaned up and textured they could be finally imported into Unity. In Unity the last step was to create the collision boxes for the objects. This is necessary because it is a huge waste of performance when resources in the final game have mesh collider which are perfectly representing object. A good option is to use primitive colliders as much as possible. This is especially true for developing mobile games with alternative input modalities like orientation to allow tilting.

Save the Bees

Tina Truong (HdM), Simo Rodbl (HdM), Sarah Lamarque (LISAA)Sophie Schwarz (LISAA)

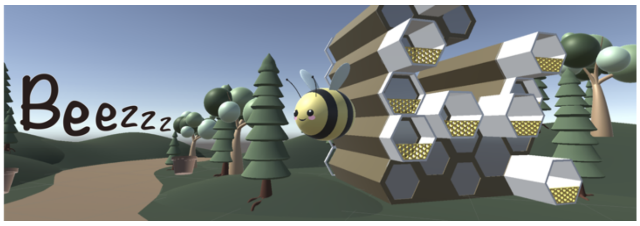


Figure 11 Tower Defence Game Save the Bees

Save the Bees (Figure 11) is a tower defence game developed in Unity, intended for web and mobile. The player's role is to protect a beehive from evil spiders. For this purpose, the player buys flowers with honey to increase the bee population in strategic locations in order to strengthen the defence. If the hive is destroyed because too many spiders reach it, the game is lost. The player wins if the hive has not been destroyed before all the spiders have passed or have been defeated. The units and environments are rendered in 3D, but the game itself is limited to a static top-down view. The orthographic camera uses a 60° angle to simplify control while still showing more than one side of the model. Since the graphic style is meant to be cute and cartoony, the models and colors are deliberately kept simple.

The original project, figure 12 is based on a multiple and individual habitat taking up the alveolar forms of the beehive, thus elaborating an infinite variety of formal proposals to recreate a community ensemble.

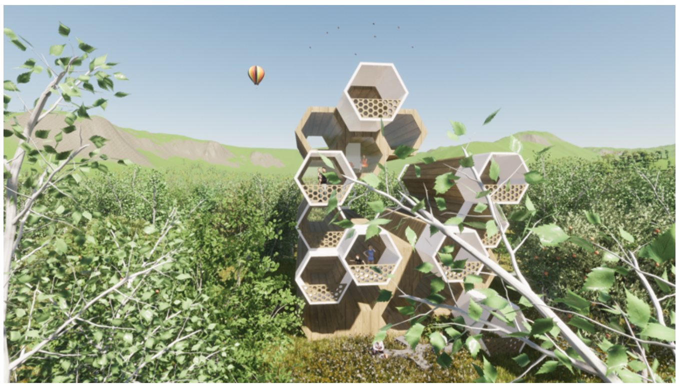


Figure 12 Rhino 3D Model

Save the Bees Data Transfer

The biggest initial question was how to best get the models from Rhino into Unity. Both Rhino and Unity have a number of commonly supported export/import formats, such as .obj and .fbx. First, we decided to do research on potentially already existing solutions for Rhino's default export format as we anticipated some problems occurring with non-default exports.

"Unity Reflect" is a Unity product explicitly aimed at making architectural visualizations work in Unity[9]. It also explicitly includes support for Rhino files[10]. Reflect comes in two forms, "Reflect Review" and "Reflect Develop". The former is intended to be used as an end product, allowing architects to showcase their models in a viewer application. The latter allows developers to further tweak this viewer application to fit their own needs. We've assessed "Reflect Develop" with a trial account but found out that the work required to tweak the viewer application would be outside the scope of development work feasible for our team size and time frame.

We've also looked at potential open-source projects, of which we found one named "Unify" that aimed to "create a streamlined pipeline from Rhino to Unity"[11]. This would be done in a way similar to how Unity handles proprietary formats in that a converter is written which turns custom formats into formats recognized by Unity - in this case, turning a Rhino file into an equivalent .obj file. Sadly, the project appears to be abandoned and incomplete and thus unusable for our purposes.

After discussing our problems with the other groups and keeping the limited time frame in mind, we decided to settle on .fbx as an export format, as it seemed to promise the quickest solution with the highest chance of success.

Importing .fbx into Unity revealed faulty geometry, missing materials, meshes that were unnecessarily dense and thus had a detrimental performance impact, misplaced object origin points and the unwanted inclusion of usually excluded scene objects such as cameras and lights. None of these were issues the LISAA students were aware of or had encountered when using Rhino before. This resulted in a steep learning curve and lots of setbacks throughout the week, as not all of the problems were recognized from the start and many required a thorough trial and error process

In detail, faulty geometry meant that meshes had normal in wrong directions and intersecting geometry, both of which produced unwanted visual artefacts. As Rhino offered no concrete options to solve these issues before export, our team tried to address them afterwards as best as we could. One solution would have been to fix up the models in one of the typically used applications for 3D game production, such as Blender or the Autodesk product family. Given our team size and the unfamiliarity of the LISAA students with these tools, this was infeasible to do in the given time span. Instead we chose to conceal and hide these issues as best as we could with clever placement inside our game scene. On the flip side, the issues with unwanted objects and origin points were possible to be fixed both before and after export in Rhino and Unity respectively and were mostly related to LISAA students' inexperience with creating assets for games. In a similar vein, materials were simply recreated in Unity and assigned to the models, which - while being an unfortunate extra step - solved the issue of missing materials.

The biggest issue that remained unsolved was the mesh density, as our project's target platforms had potentially very limited computing resources which clashed with the high vertex count. Even on PC we encountered slowdowns into single-digit FPS. Therefore, we had to drastically reduce the number of objects placed in our scene. A potential solution could have been a mesh simplification step, which other game engines such as Unreal and other creation tools such as Blender would both have supported. As we had already done substantial work, an engine switch was out of question at the time those problems were encountered. As outlined before, the use of other creation tools was also infeasible given the unfamiliarity with the tools and the limited time frame. Though numerous, none of the issues were big enough of a hurdle to prevent us from making a game. As such, we would still consider the week a success in incorporating architectural workflows into a game pipeline. We are confident in saying that given enough time, the existing issues could have been solved in a more optimal way, providing a viable pipeline for bigger productions.

In the Desert

[9] https://unity.com/de/products/unity-reflect
[10] https://unity.com/de/pages/unity-reflect-rhino
[11] https://github.com/lelandjobson/Unify

Vanessa Voge (HdM) Lisa Staehlè (LiSAA), Coline Thomann (LiSAA)

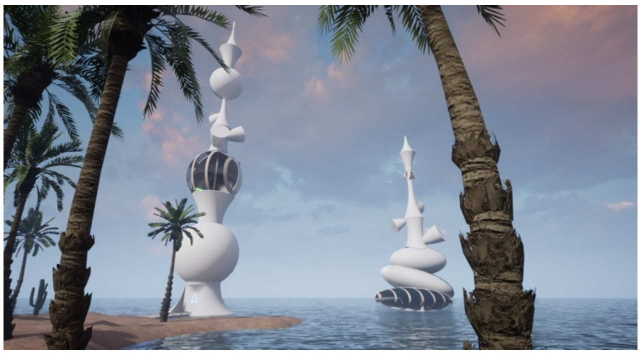


Figure 13 Rhino 3D Exercise

In the Desert (Figure 13) is a first as well as third-person exploration game with puzzle elements. It is up to the player from which perspective he or she wants to navigate through a remote island in the middle of a seemingly endless sea. The first-person perspective however is part of the main mechanic of the game: on pressing the left mouse button, a stencil overlay will appear. For simplicity basic geometric forms such as a triangles and rectangles were chosen. In order to fulfil the primary objective of climbing up the tower, the player needs to find the corresponding symbol within the environment. In concrete terms: two neighbouring palms intersect in a way that they can be viewed as a triangle. If the player looks at this from the correct perspective, fitting the stencil with what is behind it, they will be able to solve the puzzle.

For the project "In the Desert" a group of LISAA students have created two abstract, tower-like structures floating above the water near an island coast. This set of a desert island amidst a great ocean has been carried into the game, with the taller tower being accessible through a bridge. The towers were created using the modelling software Rhino 3D.

Since the students wanted to lay emphasis on the look of the environment and due to the availability of various high-quality assets suitable for world building, the Unreal Engine 4 was chosen for game development.

In the Desert Data Transfer

Figure 14 illustrates the workflow of incorporating the 3D models from Rhino 3D into the game scene in Unreal Engine.

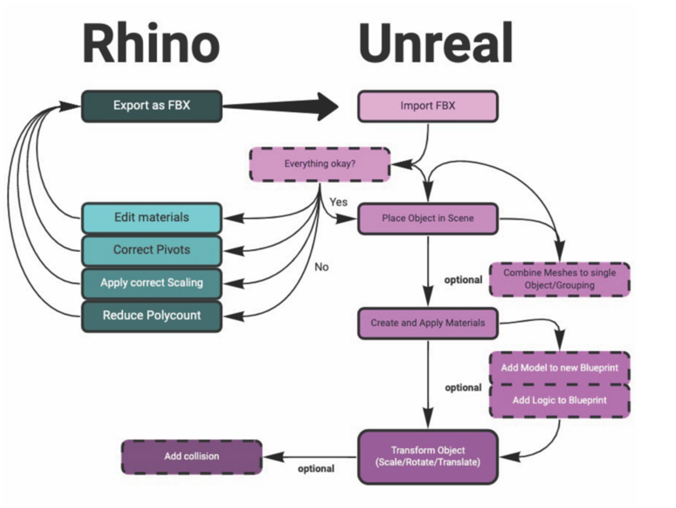


Figure 14 Data transfer workflow

Since the game engine requires 3D models to be in .fbx format, they had to be exported accordingly. After importing them into the Unreal Engine, it became obvious that based on the various roundish parts of the towers, the triangle count was very high. In terms of a game development approach, it is recommended to keep the triangle count low to ensure good performance. Thus, the models were re-exported with a lower triangle setting in Rhino. However, while drawing samples of the tower parts within the game engine, the triangle count did not seem to decrease remarkably. As the game was not intended to run on mobile, the reduction of triangles wasn't as urgent as for other projects and with the little time in mind, the group carried on. The 3D models were imported as a collection of submeshes that piece together the respective tower in the game scene. Each submesh is handled as a single object in the game engine, which allows for individual changes of certain parts, but can make the world outliner (where every object in the level is shown in a list) confusing.

What became apparent early on would turn out as the main issue of the Rhino 3D to Unreal Engine workflow: the pivot. Every submesh was relatively in the right place, but since initially both towers were placed together in the Rhino scene, the pivot lay between them. They were exported individually, but kept the askew pivot. This made any transformation very difficult, especially rotations. Numerous attempts of fixing the pivot in Rhino 3D did not solve the issue, thus a workaround in the game engine was used. To pool the submeshes into a new, single mesh or grouping them for once cleaned up the world outliner and somewhat simplified transformations of the model, as a new centered pivot is set by default. However, this proved to be unreliable in some instances, as the pivot would sometimes jump back into unfavourable places, making the whole process of placing the models into the scene quite time-consuming. Thus, a proposal for further likewise projects would be to better prepare the students to game development-friendly 3D modelling (fewer triangles, centered pivot).

Originally, the towers were hollow, so several iterations were required for the player to be able to enter and climb up the taller tower. Bit by bit, a door arch, floors and stairs (inside and outside) were added, as well as the adjustment of sizes. Smaller corrections were made preferably in the game engine itself to prevent unnecessary hassles with the pivot.

Another step in the pipeline was to add collision to relevant parts of the tower. In the import process of the Unreal Engine automatic collision can be enabled, but due to the complexity of the tower model, this led to it being inaccessible. This feature only turned out to be useful for the stairs that were handled independently from the overarching model. To add collision to the remaining parts, they had to be edited one by one in Unreal's Static *Mesh Editor*. That's why grouping the submeshes first, then merging them into a single mesh once the collision has been set, turned out to be a good approach. In respect of the mesh complexity, with the tower consisting of many crescent-shaped forms, some parameters of the *Auto Convex Collision* had to be adjusted. This resulted in collision meshes with more vertices, which can affect performance if being overdone.

If the tower would have been multi-colored, it could have led to more problems, as the import log would throw warnings about missing smoothing group information of some parts or about degenerate tangent bases, nearly zero normals, tangents and bi-normals. All of these could lead to issues such as incorrect shading. The emitting glowing material was created in the game engine's *Material Editor*, since the developer had foreknowledge on how to realize this quick and easy. The diamond, which has to be collected at the very top of the tower, threw errors and failed to import. This was caused by the translucent material, which too was created in Rhino 3D. After some attempts to fix this issue, the game engine would still file the same error message, that is unable to triangulate the mesh. Eventually, the diamond was imported with the white base material, while a new translucent material was created in the Material Editor. This was applied to the diamond, resulting in the desired look of a red, reflective gem.

Conclusion

Collaborating in an interdisciplinary and international program to challenge and discuss new developments in the 3D product design turned out to be an interesting and very productive endeavour. Focusing on transferring an architectural exercise into a game turned out to not only provide a suitable goal for students to explore but also helped to foster interdisciplinary exchange among the students and the involved faculty alike.

The necessary workflow to transfer data from Rhino 3d to Unity or the Unreal Game Engines became apparent. Particularly the difference of 3D data requirements for real time interaction such as games and 3d visualization purposes only, became very apparent. Particularly for the students who created games that allowed a player to enter architectural structures, such as was the case in the project of "Escapers" and "In the Dessert". Other project used the architectural structures as inspiration such as "Save the Bees" also encountered the issues with multiple 3D Mashed in Rhino but were able to deal with them differently or did not encounter the issue because the building was not imported directly as was the case in the "Palma" project. "Palma" "Starsphere" had the added challenge of optimizing the number of polygons for mobile devices.

The involved computer science and media students involved in the project learned not only how to work in interdisciplinary and international teams but also how design students work in 3D while focusing on the overall aesthetic and visualization of the idea. Due to their double role as tutors and game engine programmers the HdM students learned how to emphasis with the design students and focus on the data transfer which is a state-of-the-art problem that a lot of industries face while transitioning to game engines for real time interactive solutions, such as the car or virtual video productions industries.

In addition to being able to work in an interdisciplinary and international way, LISAA students learned about the difference between modelling for industrial design, very detailed, high-poly modelling, to produce photorealistic visuals, plans, prototypes and modelling for games. They also discovered the game design system through the integration in game engines.

Acknowledgements

We would like to thank the International Office of the Hochschule der Medien, especially Martine Schumacher, the LISAA International Office, especially Leila Menard and Benjamin Walker, Director of LISAA Strasbourg. And to all the participating students who took part in this experimental project.

References:

[1] Barr, C. (2018). *Real-time CAD Visualization with Unreal Studio* . Von Autodesk: https://www.autodesk.com/autodesk-university/de/forge-content/au_class-urn%3Aadsk.content%3Acontent%3Ad505fbe6-6480-4fc4-8121-22e2905a1808 abgerufen

[2] Beller, M. (2015). *Herausforderung Produkt Konfigurator (Teil 1): Anforderungen an Fahrzeugkonfiguratoren*. Von doubleslash: https://blog.doubleslash.de/herausforderung-produkt-konfigurator-teil-1-anforderungen-an-fahrzeugkonfiguratoren/ abgerufen

[3] CAD Deutschland. (2020). *Was ist CAD? - Zurück zu den Grundlagen* . Von CAD Deutschland: https://cad-deutschland.de/news/371895 abgerufen

[4] Eberly, D. H. (2006). *3D Game Engine Design - A pratical approach to real-time computer graphics (second edition).* Burlington, Massachusetts: Morgan Kaufmann Publishers.

[5] Ebner, M. (2020). *Virtual Trends: BMW und Epic Games feiern langjährige technologische Partnerschaft.* Von BMW Group: https://www.press.bmwgroup.com/austria/article/detail/T0321390DE/virtual-trends:-bmw-und-epic-games-feiern-langjaehrige-technologische-partnerschaft?language=de abgerufen

[6] Frauenhofer IOSB. (2018). *Digitaler Zwilling - das Schlüsselkonzept für Industrie 4.0*. Von Frauenhofer: https://www.iosb.fraunhofer.de/de/geschaeftsfelder/automatisierung-digitalisierung/anwendungsfelder/digitaler-zwilling.html abgerufen

[7] Gillies, C. (2021). *Game Engines in der Fahrzeugentwicklung*. Von Porsche Newsroom: https://newsroom.porsche.com/de/2021/innovation/porsche-engineering-game-engines-software-23340.html abgerufen

[8] Gregory, J. (2014). *Game Engine Architecture - Second Edition, S. 11-13.* Boca Raton, Florida: CRC Press.