sciendo

Polish Cartographical Review
Vol. 52, 2020, no. 2, pp. 61–73
DOI: 10.2478/pcr-2020-0006
Received: 05.05.2020
Accepted: 09.08.2020

JACEK BERNARD MARCINIAK, HUBERT JANICKI
Warsaw University of Technology
Faculty of Geodesy and Cartography
Department of Cartography
https://orcid.org/0000-0003-3654-5187; jacek.marciniak@pw.edu.pl
https://orcid.org/0000-0002-6867-5069; janicki.hubert@gmail.com

Using a game engine to visualize the Main Building of the Warsaw University of Technology in a mobile application

Abstract. The aim of the study presented in this article is to identify and analyse the problems which arise when creating a 3D model based on two-dimensional data and its import into a game engine and then developing algorithms to automate this process. The authors decided that they would use the Unity game engine to create an application presenting the results of modelling the interior of the Main Building of the Warsaw University of Technology. The work was divided into stages in which problems related to the adopted method were identified and the automation of selected activities was suggested. The main tasks performed during the study included processing the source data into a 3D model along with the correction of errors made during this process, detailing the model by adding characteristic elements of the building's interior, and creating the so-called game scene in the Unity game engine along with the implementation of the application's behaviour. The developed software can be integrated with indoor navigation systems, and the implemented scripts can be used during the preparation of other models.

Keywords: 3D modelling, 3D visualization, game engine, Unity, building model

1. Introduction

For years, computer game developers have been building tools and programming components enabling the creation and display of attractive visualizations. A set of such tools and programs which support the design of games and their subsequent implementation is referred to as a "game engine" (J. Lee 2016). Game engines are developed independently and used to create many games. They are also available to other designers who use them to implement their own projects. Due to their effective and attractive data visualization, they are used not only in computer games, but also in systems created for the purpose of scientific research. There are numerous examples of the use of game engines in various fields of science, e.g. in medicine – a medical application for surgery planning (P. de Heras Ciechomski et al. 2012), in biology – visualizations of molecules of chemical substances for analysing their impact on living organisms (Z. Lv et al. 2013) or civil engineering – city visualization facilitating urban planning (D. Pielak et al. 2018).

The growing interest in the use of game engines in geoinformatics and cartography is related to the visualization of spatial objects in the main views of geoinformation applications. In recent years, indoor navigation systems have been developing dynamically, and thus there is a need to create increasingly better maps, including user-friendly 3D maps that facilitate orientation in a closed space (J. Chen and K.C. Clarke 2020). Creating such a map requires the acquisition of three-dimensional spatial data and the preparation of a multi-scale presentation, which is a challenge for the system creators (D. Gotlib 2019). Game engines facilitate three-dimensional data visualization, which is why they are often used in indoor navigation systems. I. Buyuksalih et al. (2017) visualized a city fragment with buildings saved in the

CityGML[1] format. K. Liu (2017) created an indoor navigation system displaying 3D models automatically generated based on two-dimensional floor plans saved in the SVG vector format. P.K.V. Jayananda et al. (2018) and T. Rustagi et al. (2018) developed an indoor navigation systems using augmented reality, which were also created using game engines.

Preparing a 3D building visualization requires acquisition of spatial data and planning of the modelling process. Part of new buildings are designed using BIM (Building Information Modeling) systems, thanks to which it is possible to automatically generate a three-dimensional model and use it in a game engine. This approach was used in research conducted by Y. Xiong et al. (2018) and W. Natephra et al. (2017). M. Johansson (2015) additionally analyses the performance of an application for various types of buildings, with varying details, and discusses the typical challenges of BIM data modelling in order to smoothly work with the model.

Unfortunately, BIM data are not available for most buildings and to develop a 3D model it is necessary to create it from scratch. An exact model of a building can be obtained in the laser scanning process, however, as shown by the research of X. Xiong et al. (2013) and M. Kedzierski and A. Fryskowska (2015), this process requires advanced point cloud processing algorithms and the detection of individual objects. An alternative approach is to create a three-dimensional building model based on architectural and construction drawings (X. Yin et al. 2008, L. Gimenez 2015) or a hybrid approach, in which laser scanning is used to fill missing objects (K. Khoshelham and L. Díaz-Vilariño 2014).

In the presented study, the authors decided to create a 3D model of a building based on spatial data obtained from the vectorization of raster architectural and construction drawings. The expected effect was the creation of a mobile application allowing a virtual walk inside the building in which the user moves around the model and the application displays his field of view. Therefore, it was necessary to choose an appropriate game engine to which the model could be imported and easily used to create a geoinformation mobile application. The aim of the study was to identify and analyse problems which arise when creating a 3D model based on two-dimensional data and its importing into the game engine, and then developing algorithms automatically eliminating some errors related to the adopted method.

The study was carried out for the Main Building of the Warsaw University of Technology. The building is a historic architectural structure built at the turn of the 19th and 20th centuries. It has four floors with the Main Hall in the central part and a complex system of high corridors between two wings of the building (A.A. Wagner 2001). Thanks to its heterogeneity and complex structure, it is an interesting object and a good test field for various systems operating indoors, including navigation.

2. Selection of a game engine

Typical game engines offer tools for designing animations, visual effects, rendering 2D and 3D objects, programming interactions between objects according to the laws of physics, as well as the use of artificial intelligence algorithms. The game is implemented using scripts that control the behaviour of a scene[2] and handle specific events. Game engines also provide graphics editing tools for a game scene, sound, and scripts. The game design toolkit often resembles an integrated development environment (B. Cowan and B. Kapralos 2014).

The game engine for creating the application assumed in the study had to be chosen at the beginning, because the problems related to the preparation of the model depend on the selected tool. The authors considered only free game engines. There are many such engines, but according to numerous comparisons, the two most popular are Unreal Engine and Unity [2]. Based on the analysis of materials [3] and [4], the authors concluded that both engines would enable the development of an application with the desired functionality. They provide similar functions and automatically generate applications for Android, iOS, Windows and OSX systems. Ultimately, the authors decided to choose the Unity engine on the basis of a short test – developing a simple game, consisting in moving

[1] CityGML is an open XML-based spatial data format introduced by the Open Spatial Consortium [1].

[2] A game scene is a collection of objects and additional elements, such as light sources, creating a virtual world in the game.

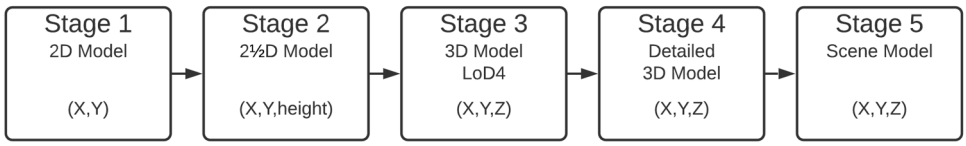

Fig. 1. The process of creating a 3D model for visualization in the game engine from two-dimensional source data

a cubic block along an obstacle course with the keyboard arrows. In the case of Unity, the implementation of the task took much less time, thanks to easier tools and the fact that the authors had better knowledge of C# used in Unity than C++ used in the Unreal Engine.

Unity uses a custom graphics rendering engine with the nVidia PhysX[3] physics engine. The scene editor enables the drag-and-drop functionality. The objects added to the project are organized in the treelike structure, and the behaviour of each object can be programmed by scripts created in C#, JavaScript, or Boo assigned to it. Unity Technologies (the developer of Unity) provides full technical documentation [5] with numerous examples and runs a forum where all users can get support from both the official technical support team and other, more experienced programmers [6]. Unity offers paid licences with extended functionality and free versions, provided that the revenue from the game does not exceed $ 100,000 per year [7].

3. Model creation process

The adopted in the study process of creating the model, from source data to the final effect was divided into the stages presented in figure 1. Such a division made it possible to define independent data processing activities with well-defined initial and final conditions and to control the process of creating the model at individual stages. In the following chapters of this article, the authors describe the implementation of the process.

Stage 1 is the acquisition and processing of 2D data. Its result is a model of two-dimensional data for individual floors of the building. In stage 2, the model is supplemented with information that allows to present objects in three dimensions, e.g. the height of floors, the height of doors and windows and their distance from the floor. Such a model is referred to in literature as the 2½D model, e.g. in J. Lesparre and BGH Gorte (2012).

The next step involves creating an appropriate model in which the coordinates of all points are three-dimensional. The intended effect is obtained by determining the *Z* coordinates of each point on the basis of information about the heights from stage 2. In this stage, the main body of the building is created, consisting of buildings belonging to individual floors, and some interior elements, such as partition walls, doors and windows.

Stage 4 involves detailing the interior of the building. The tasks of this stage include adding objects that were not represented in the input file, but are important in the 3D presentation (e.g. benches and chairs) or applying textures to model the appearance of walls, floors and doors. In stage 4, the level of detail inside the building and the accuracy of representing reality increase. From the point of view of the classification accuracy of building models proposed by the Open Geospatial Consortium (2012), the model created after step 3 and step 4 have the LoD4 level (on a scale of LoD0-LoD4 – Level of Details). This classification is insufficient to describe the accuracy of the models, especially in the context of modelling the interior of buildings. Unfortunately, a different classification has not yet been adopted, although several proposals have already been made, e.g. M.O. Löwner et al. (2016).

The last stage of working on the model is to adapt it to the requirements of the presentation in the game engine. Here, additional model optimizations are performed in order to increase the performance of the target application, the so-called the game scene, and the lighting of the building and mechanisms of moving inside it.

[3] A physics engine consists of algorithms that simulate physical phenomena in the virtual world, e.g. gravitational attraction.

An important feature of the proposed division into stages is that in each of them an independent model is created, which can be visualized, verified, and used for specific purposes.

4. 2D source model of the building

The study used spatial data acquired as part of scientific projects carried out at the Department of Cartography of the Warsaw University of Technology. This data was acquired as a result of the vectorization of architectural and construction drawings, and then verified and updated on the basis of measurements of objects indoors. The spatial data was saved in the PL-2000 coordinate system to facilitate its updating and export to applications using local rectangular reference systems.

The data model for the topography of the building is based on ESRI's BISDM 3.0 (Building Interior Space Data Model). It is a vector data model designed to represent the building, its floors and space in 2D form. Individual objects (e.g. walls) are given appropriate attributes, including room height (S. Amirebrahimi et al. 2016), which makes it easier to create a 3D model later.

The source model contained four classes of features describing the topography of the building: 1) IEA (Indoor Element Area), representing walls, doors and windows; 2) II (Indoor Installation), the connections between floors, e.g. stairs; 3) IS (Interior Space) room spaces; and 4) ISF (Interior Space Floors). The objects had an attribute denoting the storey to which they were assigned. In the case of the Main Building, nine storeys were distinguished, corresponding to the storeys of the building, and the mezzanines with the library levels and the entrance hall. Walls and doors also had an attribute specifying their height, and windows had information about their distance from the floor. The source model can therefore be described as 2½D.

5. Creating a 3D model of the building

a. GIS software used in 3D modelling

Three programs were taken into account for the work on the 3D model: ESRI CityEngine, SketchUp by Trimble, and Blender (freeware). The drawback of SketchUp is that it does not natively support the FBX[4] data format. The choice between the other two was made on the basis of tests. In both applications, the authors created a simple 3D model of a single-family house, including doors, windows and textures applied to all of objects. Navigating in both apps was intuitive, but in the CityEngine creating and texturing the model was much faster. Moreover, the speed of importing the source model and program performance during its edition were compared. Similar results were achieved by both applications, navigating the model was smooth and trouble-free. Ultimately, the authors decided to use the CityEngine.

ESRI CityEngine is advanced 3D modelling software. An innovative feature of CityEngine is the so-called procedural approach to creating a model consisting in describing the model through a sequence of commands defining the steps of creating geometric solids [8]. In addition to the typical functions, the software allows the model to be placed on the website provided by ESRI. This solution makes it possible to present the results of your work using a web browser (S. Pal Singh et al. 2014).

b. Preparation of the model

The three-dimensional building model was developed from the source model described in chapter 4 using the ESRI CityEngine software. Then, it was exported to the FBX format, which is a popular file format used to exchange 3D models between graphics design programs. The adopted approach was to create three-dimensional models of individual floors from 2½D data and then combine them into a single building model. This process may have been partially automated, but it was the source of many errors, such as not matching the shape of objects on adjacent floors or incorrect heights of some doors and windows, requiring manual adjustment.

The division of objects related to the source model classes was insufficient for effective work. Therefore, a division was made taking into account more types of objects present in the modelled building: walls, stairs, floors, windows, doors, room ceilings, furniture, railings, the external glass elevator and the ground repre-

[4] FBX is a data format implemented by AutoDesk, used to import and export 3D models.

senting two interior courtyards of the Main Building. Each object was assigned an attribute specifying its type. This made it easier to work with the model in CityEngine, because based on the value of the attribute, layers were defined, the visibility of which could be turned off while working with the model. This division was also used when automatically applying textures to all objects in a given layer and when exporting the model to the game engine.

c. Correction of door heights

Some of the objects representing the doors were exported with an incorrect height, moreover, the objects above them did not completely fill the plane of the wall (fig. 2A). The problem was that all doors were automatically assigned a fixed height in the source model, while some of them were non-standard. Therefore, it was necessary to review all the doors in the building to find their actual height, and move the wall over their top edge. The results of the corrections are presented in figure 2B. The authors attempted to automate this process, but due to the fact that each space had a different size, it was impossible to implement a universal script and a time-consuming modification of the model was necessary.

d. Filling in the missing parts of the staircase in the Main Hall

The staircase modelling in the Main Hall was also a time-consuming process. In the source model, not all the objects constituting it were present, there were no pillars supporting the successive levels of stairs. Moreover, it was necessary to correct the geometry of the objects making up the staircase in order to adjust them to each other and to the adjacent walls and floors. The results of the corrections are presented in figure 3. This problem is related to the adopted method of 3D model creation. In the source data, successive storeys contained two-dimensional objects without depicting several objects one above the other. For this reason, the pillars above the stair railings and some other objects were omitted.

e. Matching adjacent floors

The biggest problem with the adopted model creation process were the differences in shapes

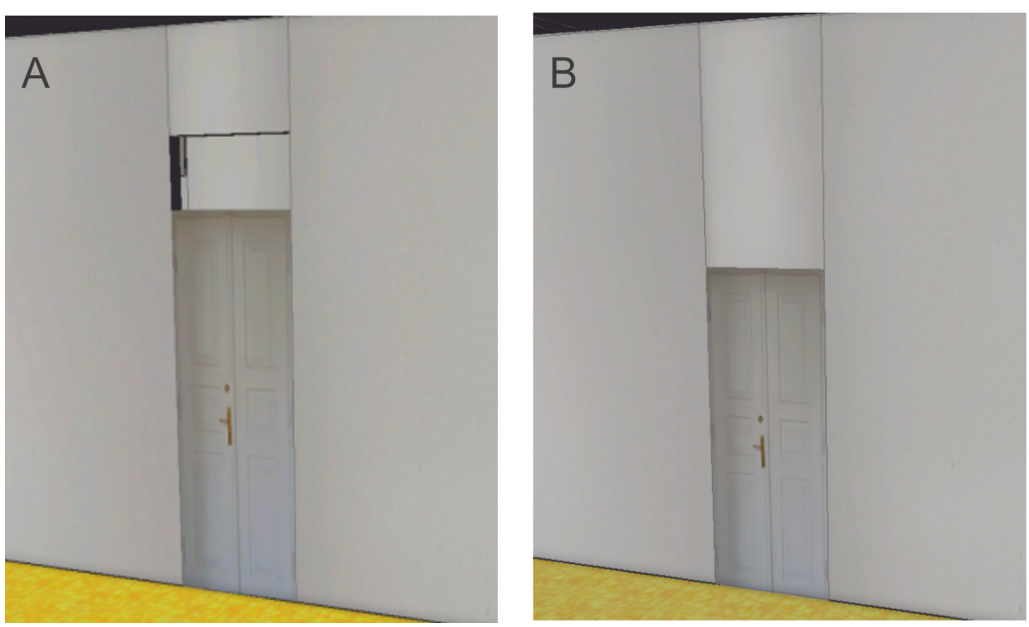

Fig. 2. Correction of the door height: A) before the change, B) after the change

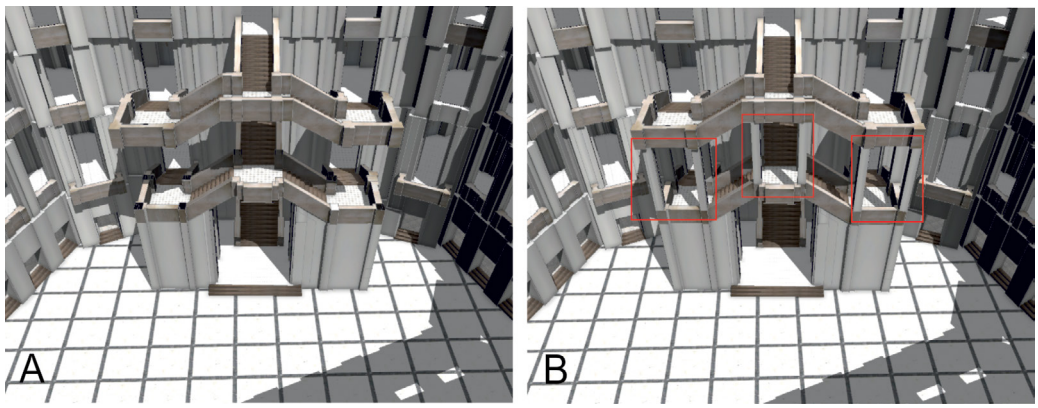

Fig. 3. The appearance of the staircase: A) before the corrections, B) after adding the pillars

of adjacent objects on different floors. The effect can be observed when the floors are connected by stairs, which in the model were part of the lower storey, and their top often did not match the shape of the upper story. The problem required manual geometry correction for all stairs.

A similar situation occurred at joints of columns on the border of the floors (fig. 4). After analysing the source materials, the authors concluded that the discrepancies between the storeys were due to several factors:

• structural elements of the building, which are in fact tilted or crooked, were shown as straight from their outline near the floor,

• the 2D model was created from low-quality architectural and construction materials,

the complex geometry of the decorative columns was generalized for each level separately,

• without taking into account the topological relationships between the floors.

Ultimately, the problem was not completely eliminated due to the large number of correc-

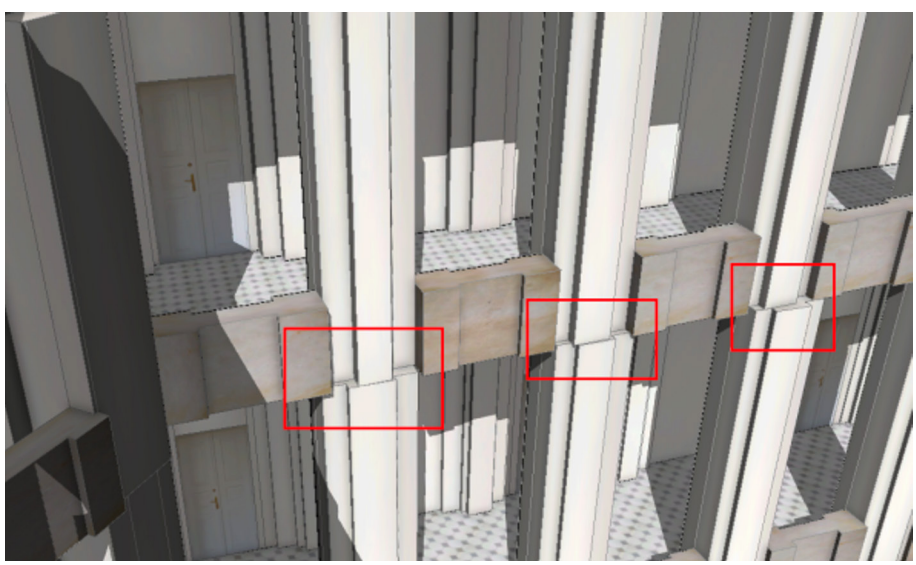

Fig. 4. Inaccurate alignment of the shapes of objects on adjacent floors

tions needed. Continuing the study, the authors plan to automate the process and outline the columns the same way for all storeys.

6. Refinement of the 3D model

a. Applying textures

Applying textures is a key part of Stage 4. The user can compare the pattern or characteristic features of the appearance of objects, which makes it easier to identify them in space. With the help of textures, it is also possible to simulate small objects, such as handles, knobs or decorations.

Textures were applied to all objects inside the building, including floors, walls, windows, etc. To this end, it was necessary to acquire photos of the building taken under good lighting conditions. Then, the photos were transformed into textures using CityEngine's built-in tool. It allows to edit textures, including to transform photos into textures, taking into account the conversion from central projection to orthogonal projection using the Helmert conformal transformation. Thanks to the division into layers, the textures were applied automatically to certain objects (e.g. windows, walls and doors). In the case of floors, the application of textures required manual adjustment due to the variety of texture patterns (e.g. wooden parquet).

b. Modelling the stairs

In the source model, stairs that connect different storeys are represented by rectangles. The three-dimensional model of the building with such stairs looks very unnatural, as the stairs resemble ramps and distract the user's attention (fig. 5A).

Attempts to solve the problem by applying textures did not give satisfactory results and it was necessary to modify the stairs by creating individual steps. The authors implemented a script that automatically generated steps based on the height of the stairs. In the next step, the script applied textures taking into account different patterns on the top, bottom and side surfaces of each step. An example of the appearance of stairs generated with the script is presented in figure 5B.

c. Adding room numbers on the doors

An important element that facilitates moving around the building are the numbers on the doors. Initially, the authors tried to faithfully reproduce the actual location and size of the

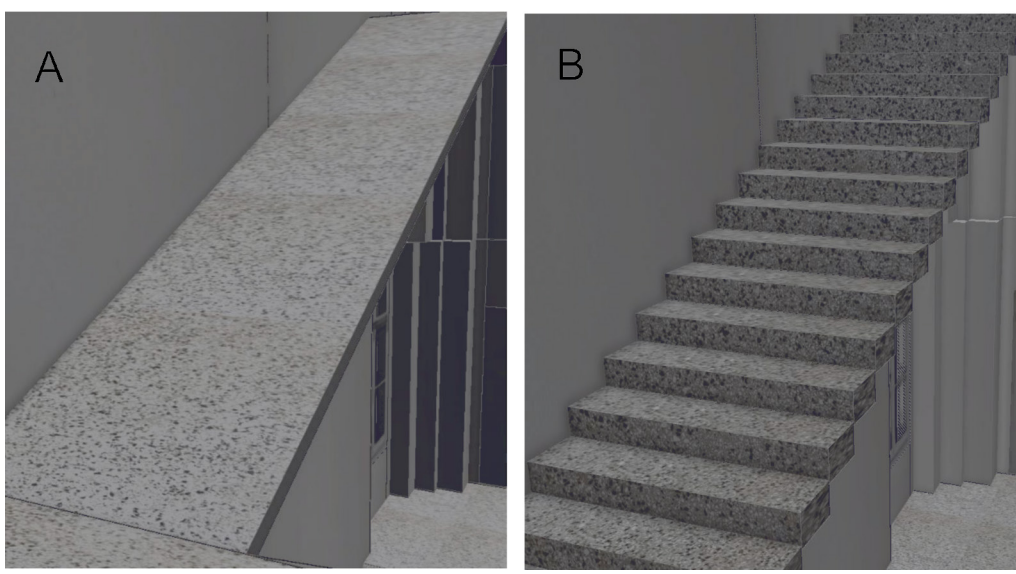

Fig. 5. Stairs A) before and B) after applying the modelling script

digits, but the model prepared in this way made it impossible to read the number while virtually moving along the corridors. For this reason, it was decided to put enlarged room numbers on the doors, slightly lower than in reality, making them easily noticeable from a distance and at a slight angle[5] (fig. 6).

Fig. 6. Door with room number

d. Adding interior elements

As part of the next task, interior elements were added to the model. Due to the size of the facility and the presence of numerous frescoes, paintings, commemorative plaques and monuments, the authors decided to prepare the interior of lecture room 315 as a representative fragment of the model.

Room 315 can accommodate about 200 people, and its characteristic feature is a staircase arrangement of individual levels with benches. The floor of the room is divided into thirteen levels, of which twelve are benches. In order to obtain models of equipment objects, the authors made manual measurements of the room with accuracy to 1 cm using a tape measure. In this way, the width and height of the stairs and benches, the distance between the benches and the walls, the width of the central passage between the benches, the dimensions of the platform for the teacher and the dimensions of the board located in the centre of the room were acquired. Using the PL-2000 reference system in the model facilitated the process of placing objects on the basis of the measured distance. The results of the work are presented in figure 7.

7. Preparing the model for display in the Unity

a. Creating a scene in the Unity

After modelling the spatial data, the model was imported into the Unity scene editor. Data exchange between CityEngine and Unity was performed using the FBX format.

After importing the model, the reference system was changed to move the point (0.0) to the centre of the building. Thanks to this, the game engine can more efficiently and accurately perform calculations on numbers with a small absolute value. On the other hand, it does not affect the possibility of integration with other systems, because it is easy to convert the coordinates to the PL-2000 reference system.

b. Adding light sources to the scene

When creating the game scene, the next step was to add the appropriate light sources. In the case of the visualization of the Main Building of the Warsaw University of Technology, the authors made an attempt to faithfully reproduce the lighting coming from the lamps in the building.

The authors experimented with various types of lighting and finally decided to reject the choice of spot light sources imitating lamps in favour of point light sources in order to speed up the application. The arrangement of point lights gives a similar effect to spot lights while maintaining the smooth operation of the application (fig. 8).

c. Adding colliders

A collider is an invisible and impenetrable surface that, when applied to objects, prevents

[5] Due to the time-consuming nature of the process, the authors decided to put the numbers on the doors only in selected corridors in the prototype application.

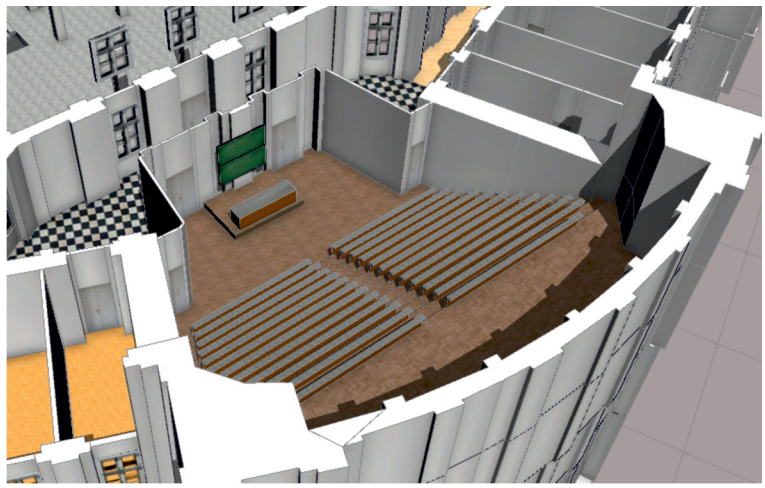

Fig. 7. The appearance of lecture hall 315 (the ceiling has been removed for better visibility)

them from penetrating one another. Thanks to this, it is possible to introduce a virtual character moving through the scene, who cannot move through the walls. In the case of the application created by the authors, a character means a virtual, invisible person whose field of view is displayed on the device screen. In designing the movement of the building, colliders were added to the walls and floors, leaving the possibility of moving through corridors and entering selected rooms through doors.

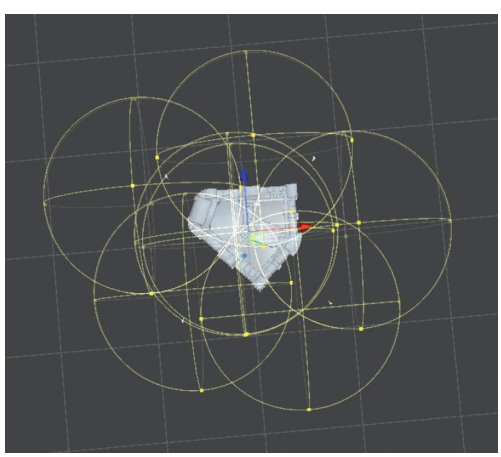

Fig. 8. Arrangement of point lights on the scene, top view

d. Navigation around the building

The Unity game engine provides a component called "*RigidBodyFPSController*" containing a set of methods that support the movement of characters in a virtual world using keyboard arrows and computer mouse movements. Due to the fact that the application was to be run on mobile devices with touch screens, it was necessary to implement own mobility mechanisms. For this purpose, three controls were added to the screen (fig. 9):

- joystick (no.1) responsible for moving forward, backward, left and right,
- button (no. 2) responsible for jumping,
- invisible panel (no. 3) changing the direction in which the camera is pointing.

An interesting mechanism used in the application is the possibility of using elevators. If the user enters an elevator, a dialog box is displayed to select the floor to which they will be transferred (fig. 10).

An additional facilitation when moving inside the building is the ability to quickly move to defined places. When the user clicks the "Menu" button located in the upper left corner of the window, a dialog box appears with a list of places to which the user can move (fig. 11).

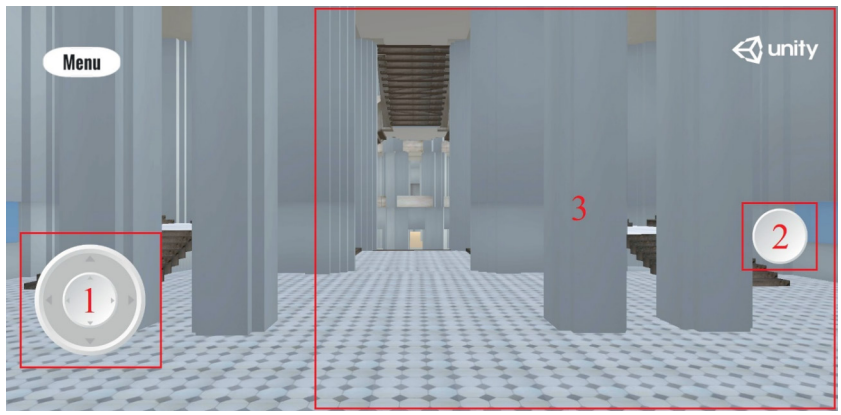

Fig. 9. Navigation in the application

Fig. 10. Floor selection panel in the elevator

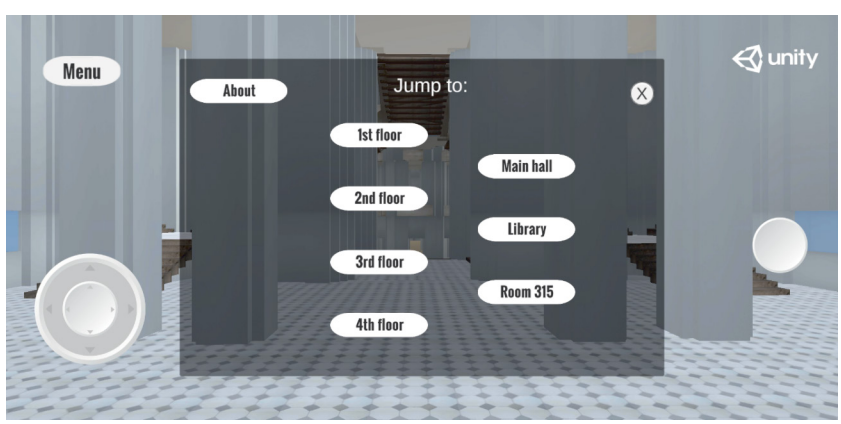

Fig. 11. List of defined places in the building, to which you can quickly move, displayed after clicking the “Menu” button

e. Optimization of the mobile application

In the process of developing computer games, it is important to optimize the game scene in order to obtain high performance of the application. Performance can be measured as CPU load, GPU load, or RAM usage. However, in the case of games, the most commonly measured parameter is FPS (Frames Per Second). 30 FPS is the limit value below which the animation is not smooth.

After importing the model from CityEngine and creating the game scene, the application was very slow. A scene analysis allowed to identify potential causes of the problem. The data model consisted of approximately 26,000 objects that the Unity engine rendered separately. Moreover, the scene contained over 100 spot lights simulating the actual lighting of the building, but significantly increasing the complexity of calculations.

The authors conducted an analysis that allowed to confirm the causes of the problem and prepare an optimized data model. For this purpose, the application was prepared for testing by placing the FPS counter on the main screen and adding the function of saving this value to a text file every 1 second.

4 building models were prepared for testing:

• Model 1 with 26,000 objects and over 100 light sources.

• Model 2 in which the number of objects was reduced to 500, combining objects with the same texture. The “Merge meshes by material” option was used when exporting the model in CityEngine.

• Model 3 with optimized light sources. Instead of 100 spot lights, 6 point sources were used to obtain similar illumination of the scene.

• Model 4 containing both the optimized number of objects and light sources.

Each model was tested in the application, collecting FPS readings as the user moves along the four paths. The paths are designed in such a way that they pass through various characteristic parts of the model:

• Path 1 along the corridors on the fourth floor – a small number of objects visible on the screen at the same time.

• Path 2 passing through the Main Hall – a large, open space.

• Path 3 going up the stairs – steps created automatically by scripts.

• Path 4 inside room 315 – a fragment of the model with the greatest detail.

FPS measurements were taken every second and lasted approximately one minute. Measurements started after waiting ten seconds after starting the application to eliminate the influence of other factors, such as the time to load the entire model into memory. Table 1 shows the mean FPS values determined in ten tests for each of the four models and each of the four test paths.

Table 1. Average FPS values for four paths and four models; 10 tests were performed for each scenario

	Path 1	**Path 2**	**Path 3**	**Path 4**
Model 1	8.1	6.9	6.4	7.2
Model 2	14.8	18.7	13.3	19.5
Model 3	18.4	13.6	15.1	12.3
Model 4	52.6	51.6	52.1	52.5

The performed study confirmed that in order to obtain smooth operation of the application, it was necessary to optimize the model by significantly reducing the number of objects. In addition, it was important to reduce the number of light sources placed on the scene. Taking both optimizations into account allowed for a smooth display of the model with over 50 FPS.

When comparing the test results of the models for different test paths, significant differences for models 2 and 3 could be noticed. The FPS values clearly depend on the test path – on the number of displayed objects for model 2 and the number of visible light sources for model 3. For model 4, the differences in FPS for different paths are small, and therefore this model did not require further optimization.

8. Summary and conclusions

As part of the study, it was possible to model the Main Building of the Warsaw University of Technology and to implement an application that enables a virtual walk around the building, which the authors plan to make available in Google Play. The use of the Unity game engine

facilitated the creation of a mobile application and enabled an attractive presentation of a three--dimensional building model. The application can be used in the future to visualize other models, but it can also be used as a component in indoor navigation systems.

As the study showed, it is possible to create a three-dimensional building model based on 2D source data for individual floors, although the model creation process requires modifying objects that cannot be represented as two-dimensional data. During the work, there were problems related to modelling 3D buildings, creating mobile applications displaying three--dimensional models, and problems closely related to the selected approach to the project. In line with the aim of the study, the problems were identified and described, and many of them were solved with implemented scripts automating the work. Unfortunately, not all tasks could be accelerated and some corrections in the data had to be done manually, making the process more time-consuming. Many of the solved problems are typical for the chosen method of creating a 3D model of the building, so the developed scripts can be reused in the future.

Continuing the research, the authors plan to improve the building model by eliminating the remaining errors in the geometry of the objects and modelling additional details in the corridors and the Main Hall. The possibility of using the obtained visualization for indoor navigation opens up new research directions related to 3D cartographic presentation.

Acknowledgements

The authors would like to express special thanks to Miłosz Gnat for valuable advice related to the selection of IT tools and help in solving several key problems.

The research described in this article is based in part on the results of Hubert Janicki's engineering thesis under the supervision of Jacek Bernard Marciniak, defended in February 2020.

Literature

Amirebrahimi S., Rajabifard A., Sabri S., Mendis P., 2016, *Spatial information in support of 3D flood damage assessment of buildings at micro level: a review.* In: 11th 3D Geoinfo Conference, Athens.

Buyuksalih I., Bayburt S., Buyuksalih G., Baskaraca A.P., Karim H., Rahman A.A., 2017, *3D modelling and visualization based on the unity game engine – advantages and challenges*. "ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences" Vol. 4, no. 161.

Chen J., Clarke K.C., 2020, *Indoor cartography*. "Cartography and Geographic Information Science" Vol. 47, no. 2, pp. 95–109.

Cowan B., Kapralos B., 2014, *An overview of serious game engines and frameworks.* In: IEEE 14th International Conference on Advanced Learning Technologies, Athens.

de Heras Ciechomski P.M., Garcia J., Olariu R., Dindoyal I., Le Huu S., Reyes M., 2012, *Development and implementation of a web-enabled 3D consultation tool for breast augmentation surgery based on 3D-image reconstruction of 2D pictures*. "Journal of Medical Internet Research" Vol. 14, no. 1, p. 21.

Gimenez L., Hippolyte J.L., Robert S., Suard F., Zreik K., 2015, *Review: reconstruction of 3D building information models from 2D scanned plans*. "Journal of Building Engineering" Vol. 2, pp. 24–35.

Gotlib D., 2019, *Selected qualities of mobile maps for indoor navigation*. "Polish Cartographical Review" Vol. 51, no. 4, pp. 155–165.

Jayananda P.K.V., Seneviratne D.H.D., Abeygunawardhana P., Dodampege L.N., Lakshani A.M.B., 2018, *Augmented reality based smart supermarket system with indoor navigation using beacon technology (asy shopping android mobile app)*. In: 2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS), pp. 1–6.

Johansson M., Roupé M., Bosch-Sijtsema P., 2015, *Real-time visualization of building information models (BIM)*. "Automation in Construction" No. 54, pp. 69–82.

Kedzierski M., Fryskowska A., 2015, *Methods of laser scanning point clouds integration in precise 3D building modelling*. "Measurement" No. 74, pp. 221–232.

Khoshelham K., Díaz-Vilariño L., 2014, *3D modelling of interior spaces: Learning the language of indoor architecture*. "The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences" Vol. 40, no. 5, p. 321.

Lee J., 2016, *Unreal Engine. Nauka pisania gier dla kreatywnych.* Gliwice: Helion.

Lesparre J., Gorte B.G.H., 2012, *Simplified 3D city models from LiDAR*. In: XXII ISPRS Congress, Commission II, Melbourne, Australia, 25 August – 1 September 2012; IAPRS XXXIX-B2. International Society for Photogrammetry and Remote Sensing.

Liu K., Motta G., Tunçer B., Abuhashish I., 2017, *A 2d and 3d indoor mapping approach for virtual navigation services*. In: 2017 IEEE Symposium on

Service-Oriented System Engineering (SOSE), pp. 102–107.

Löwner M.O., Gröger G., Benner J., Biljecki F., Nagel C., 2016, *Proposal for a new LoD and multi-representation concept for CityGML*. "ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences" Vol. 4.

Lv Z., Tek A., Da Silva F., Empereur-Mot C., Chavent M., & Baaden M., 2013, *Game on, science-how video game technology may help biologists tackle visualization challenges*. "PloS one" Vol. 8, no. 3.

Natephra W., Motamedi A., Fukuda T., Yabuki N., 2017, *Integrating building information modeling and virtual reality development engines for building indoor lighting design*. "Visualization in Engineering" Vol. 5, no. 1, pp. 1–21.

Open Geospatial Consortium, 2012, *OGC City Geography Markup Language (CityGML) encoding standard.* Tech. Rep. Nos. OGC 12-019, version 2.0.0. Wayland, MA.

Pal Sigh S., Jain K., Ravibabu Mandala V., 2014, *Image based Virtual 3D Campus modeling by using CityEngine*. "American Journal of Engineering Science and Technology Research".

Pielak D., Kowalski M., Lebiedź J., 2018, *3D model preparing patterns for interactive urban visualization*. "TASK Quarterly: Scientific Bulletin of Academic Computer Centre in Gdansk" Vol. 22, no. 4, pp. 341–349.

Rustagi T., Yoo K., 2018, *Indoor AR navigation using tilesets*. In: Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology, pp. 1–2.

Wagner A.A., 2001, *Architektura Politechniki Warszawskiej*. Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej.

Xiong X., Adan A., Akinci B., Huber D., 2013, *Automatic creation of semantically rich 3D building models from laser scanner data*. "Automation in Construction" No. 31, pp. 325–337.

Xiong Y., Bulbul T., Reichard G., 2018, *BIM and game engine integration for operational data monitoring in buildings*. In: 7th International Building Physics Conference, IBPC2018.

Yin X., Wonka P., Razdan A., 2008, *Generating 3d building models from architectural drawings: a survey*. "IEEE Computer Graphics and Applications" Vol. 29, no. 1, pp. 20–30.

Internet sources

[1] CityGML, https://www.ogc.org/standards/citygml (access 17.05.2020).

[2] The Top 10 Video Game Engines, https://www.gamedesigning.org/career/video-game-engines/ (access 9.05.2020).

[3] Unity vs Unreal: Which Engine Should You Choose As A Beginner, https://www.youtube.com/watch?v=zsL6LYVYU5c (access 20.05.2020).

[4] Unity vs Unreal, Graphics Comparison, https://www.youtube.com/watch?v=S2eXK025uC4 (access 20.05.2020).

[5] Unity User Manual, https://docs.unity3d.com/Manual/index.html (access 20.05.2020).

[6] Forum Unity, https://forum.unity.com/ (access 20.05.2020).

[7] Unity, Plans and pricing, https://store.unity.com/ (access 13.04.2020).

[8] ESRI CityEngine, https://www.esri.com/en-us/arcgis/products/esri-cityengine/overview (access 13.04.2020).