ISSN 1828-5961

Daniel Martín-Fuentes
Phd. Architect and Professorat the Universitat Politècnica de València in the Graphic Expression Department. He has carried out research stays at CRENAU (Nantes) and at the Università Luigi Vanvitelli (Naples) with scholarships from the Italian ministry. He focuses his research on the applications of virtual reality in the analysis, study and dissemination of architectural heritage.

Pedro M. Cabezos-Bernal
Phd. Architect and Professorat the Universitat Politècnica de València in the Graphic Expression Department. Hisresearch line is focused onnew representation tecniques such as 3D modelling, virtual reality, immersive photography, photogrammetry SfM (Structure from Motion) and ultra-high resolution photography.

Universal point cloud viewer based on Unreal video game Engine

Point clouds are an essential tool for documenting archaeological and architectural heritage, since they are the initial product that can be obtained from a laser scanner or photogrammetry survey. Despite being a very basic and discretized source of information containing only coloured points, they can be used as a first approach to the surveyed model through a virtual visualization or interactive tour.
The management and visualization of point clouds is not an easy task, since it involves the representation of millions of points and installing specific viewers is mandatory. These special viewers are developed by each laser scanner manufacturer to visualize its own point cloud file format. Handling with them can result unfriendly and non-intuitive for a non-specialized public. Therefore, the main objective of this work is to develop a working methodology based on Unreal Engine, a video game development environment, to create self-executing visualization modules that can be shared with any user. These modules will allow simple and fluid navigation of point clouds without the need of installing any additional software or having specific knowledge.
These self-executing modules will allow any kind of viewer, specialised or not, to visualize the point cloud in a first-person way or from a bird's eye view. The exposed methodology has been tested using a survey of the Oliva Castle in Valencia (Spain), which is classified as a cultural interest asset. In order to illustrate the results of this research, some images of the visualization environment will be shown and the download link of the display module corresponding to the Oliva Castle will be provided.

Keywords:
Point Cloud; Unreal; Viewer

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

INTRODUCTION

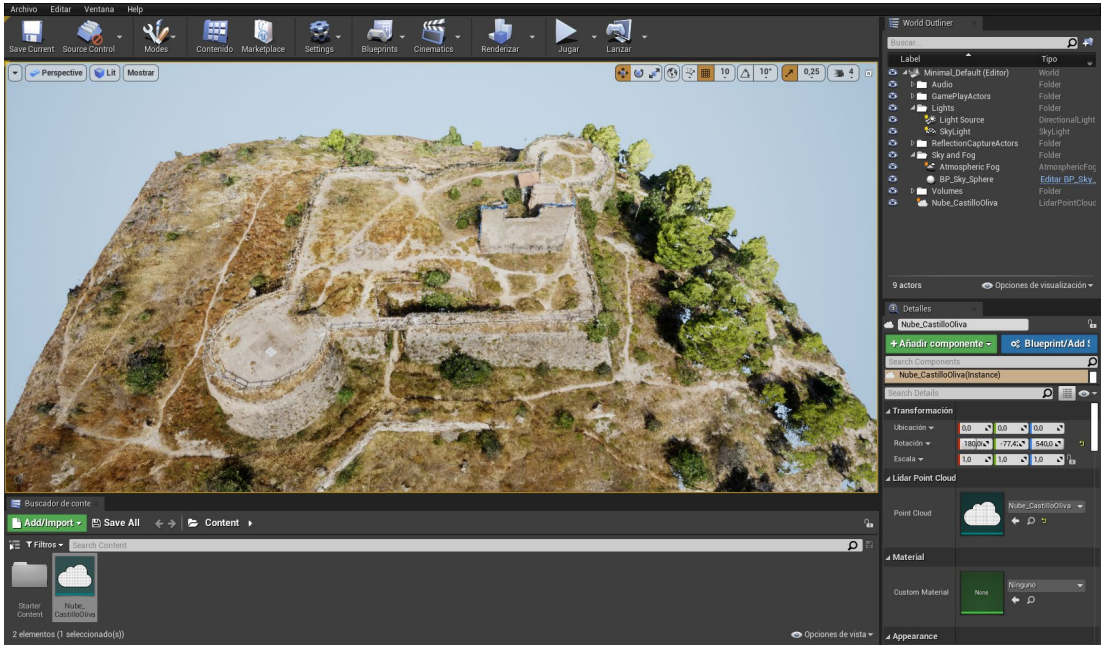
Point clouds are an essential tool for documenting archaeological and architectural heritage, since they are the primal result that can be obtained from a laser scanner or a photogrammetry survey. Despite being a very basic and discretized source of information containing only coloured points (Fig. 1), they can be used as a first approach to the surveyed model through a virtual visualization or interactive tour. The management and visualization of point clouds is not an easy task, since it involves the representation of millions of points and installing specific viewers is mandatory. Most of these special viewers are commercial software, however there are some free alternatives. Almost every Laser Scanner brand has its own free viewer. For instance, Leica provides the software Leica Cyclone 3DR Viewer [1]. Faro brand also freely supplies its software Faro Zone Viewer [2], but it is only capable to visualize some proprietary formats of the products developed by the firm.
Video game engines like Unreal Engine (UE) are excellent visualization environments as they are optimized to get the most out of the hardware, so they manage the available resources outstandingly. In addition to this, UE is free for non-commercial purposes and for project with earnings under 1 million dollars [3]. It allows creating elements that interact with the viewer, thanks to its graphic programming environment based on blueprints or even using the traditional C++ programming language. In that way, it is possible to include additional interactive information that can be used to generate a musealized virtual tour from the point cloud.

MANAGING POINT CLOUDS WITH UE

UE is a software of Epic Games. It was designed initially with the only purpose to build first person shooter games but due to its potential was later used for the development of various apps (Kontos, 2020). Built up to light, shade, animate, composite and structure virtual environments it lacks advanced modelling though. This means that most of the physical elements have to be imported and placed in the scene and then materials, lights, animations, particle systems, collisions and many other characteristics that complete a virtual world have to be applied inside UE. Working with Point clouds follow the same workflow, but they cannot be imported directly since it is necessary to use the LiDAR Point Cloud Plugin to manage them.
The plugin, that was developed by an independent anonymous programmer, was published to the Epic Games marketplace, completely free, in February 2018, coinciding with the award of an Unreal Dev Grant. In May 2020, after seven versions, it was finally built in the editor, although it has to be specifically turned on. The developer, who was very active in the Epic Games Dev Community [4], carried out a continuous improvement of the plugin with the comments received in the forum, but he stopped answering requests from February

Fig. 1 - Photogrammetry 3D model of "Donjon" in Klodzko Fortress – Source: Frankzuc et al., 2022.

Fig. 2 - Aerial view of a point cloud in UE.



DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

2022. The project seems to be frozen for now at version 0.8, so it remains the same even with the huge change that UE has done from version 4 to 5. The plugin is a beta version yet but is fairly stable for the planned research shipping.
As it is shown in the documentation of the software, the latest version of the plugin has several features [5]: It supports most of the file formats for point clouds, ASCII (TXT, XYZ, PTS) and LAS, e57 files; supports dynamic shadows, both cast and receive; renders data as pixel points or as sprites [6]; implements eye-dome lighting technique to improve shape accentuation; it has blueprint support; allows multiple coloration techniques (RGB, Intensity, Elevation, Classification); has a dynamic LOD system and GPU streaming permitting large asset loading and finally allows asynchronous data importation and modification during runtime (Fig. 2).
All these characteristics are a powerful toolset to work with point clouds and may simplify the management of large data sets when it comes to visualization. Moreover, UE5 seems to be able to render the tiniest details of a 3D model even with very high object counts with a negligible performance impact. Therefore, the main objectives of this research will be centred on developing a methodology that will allow creating and publishing virtual visits that can be fully interactive, specially focusing on UE's property of creating stand-alone apps that will run in any computer and almost any operative system without installing any software. Another outcome to not depend on using UE will be providing a fully functional independent app (built with UE) that will be able to load point clouds and may act as a powerful and useful visualization tool, which can highly simplify things at the cost of some limitations, as will be mentioned onwards.

THE LIDAR POINT CLOUD PLUGIN

As it was explained before, when creating a new UE project, it is necessary to activate the LiDAR Point Cloud Plugin in the plugins panel in order to be able to import point clouds. Since it is a beta version, it gives a warning that has to be taken into account, but after all its use during this research the plugin can be considered to be stable in everything related to visualization. On the contrary, when it comes to editing, modifying or exporting the performance turns unstable.
In the Project Settings panel (Fig. 3a), it is possible to change the octree and performance parameters, but the values are so optimized that this is not necessary. For every physical value, UE works in centimetres, so there is an import/export scaling value that transforms the units properly. Finally, the automation section has two important parameters; the first one is the "Auto centre on import", checked by default because preserving original coordinates may cause noticeable precision loss if the values are too large; the second is the "Auto build collision on import" that can be interesting in light point clouds, but can crash when importing heavy ones. In those cases, is better to build the collision in the editor panel of the asset (Fig. 3b).
When importing large point cloud files, the engine requires a significant amount of RAM in order to load data and turn the cloud into UE assets. Once the file is fully loaded, the software will keep the asset in the system memory and only load a certain number of points as needed, depending on the LOD count. This results in a reduction in RAM consumption and consequently fast asset loading. In the overview of the plugin [7], we can find the performance indicators about the load of a public-

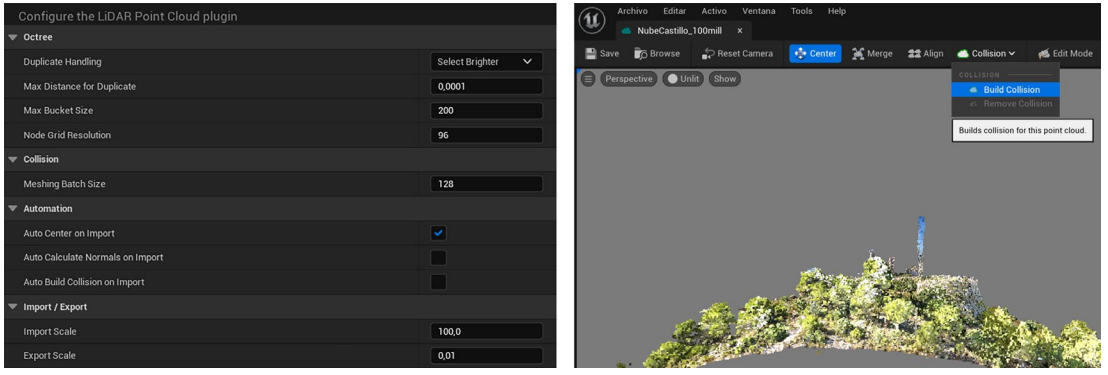
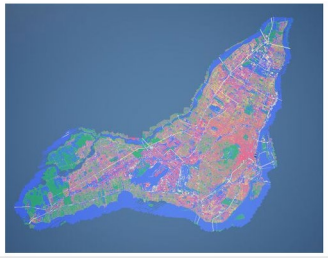


Fig. 3 - a) Plugin parameters in Project Settings – b) Build collision in Point Cloud Edit Panel

Fig. 4 - Performance indicators in a publicly available LiDAR data, Montreal. Source: docs.unrealengine.com



Individual LAS files	~684 tiles for a total of 253 GB on disk
Unreal RAM usage	~3.5 GB for a budget of 1 million points
Frame Rate	120 fps, with a global budget count of 1 million points
Total Amount of Points	Average 14.3 million points per 1 km x 1 km tile * 621 files = ~8.9 billion points

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

ly available LiDAR data from the city of Montreal (Fig. 4), but these are always ideal values achieved with very high-end computers and more realistic values will be checked during the study. It also has to be pointed out that in the example the whole model is split into 621 files of approximately 370 MB each one, which is not always the common situation. It is rather more usual to have a large unique file, mainly in the study of heritage where a global composition is generally needed to debug and organize the information.

Once the point cloud is in the scene, twenty different parameters can be read and modified, but the focus will be, in our case, on the ones that affect perception. As points are rendered as sprites their shape can be changed and scaled on demand. Circle and square shapes are the ones by default (Fig. 5), but by creating a sprite material it is possible to adopt whatever shape (Fig. 6b). When the point size is set to zero, points render as pixels (Fig. 6a) and in general square shapes fill better the gaps between points but give a more planar sensation. Other parameters such as the scaling method, the gap filling strength and the point orientation can be modified with less affection on the final image. The colour source will allow to choose the colour information channel between data, elevation, position and classification (Fig. 7). If no channel is chosen, then the colour will be the one given by the material applied to the point cloud.

In terms of lighting and shadows, that have a great impact on the quality perception of the representation, the workflow is simple. As the clouds generally come with the real colour captured in the scan, there is no need to place lights actors. This also skips the need to build lighting and thus eliminates calculation processes. In order to see the clouds with the colour and light with which they were scanned, the plugin creates the material of the sprites automatically with the colour information applied in the base and emissive channels.

Several console variables related to point clouds are created when the plugin is activated [8]. Their default values are optimized for general uses and their influence in the perception is minimal except for the variable r.LidarPointBudget. UE5 works

Fig. 5 - Point shape circle - square (both cases with Point Size = 1).

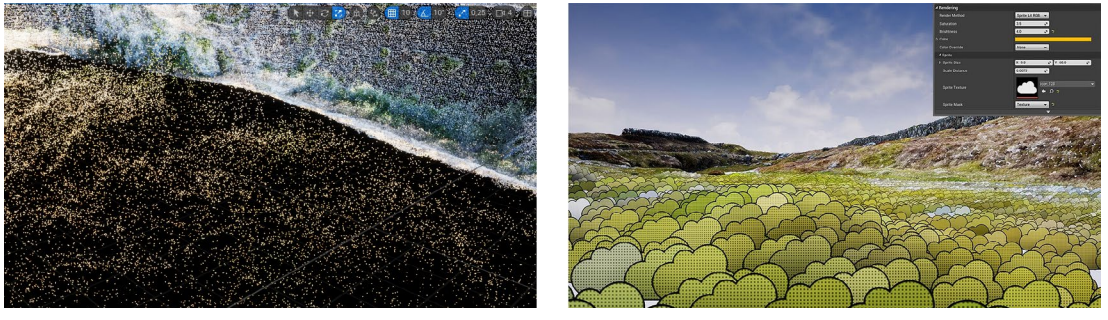


Fig. 6 - a) Point size = 0, Point as pixels – b) Sprites in the shape of a cloud. Source: Phoboz in dev.epicgames.com

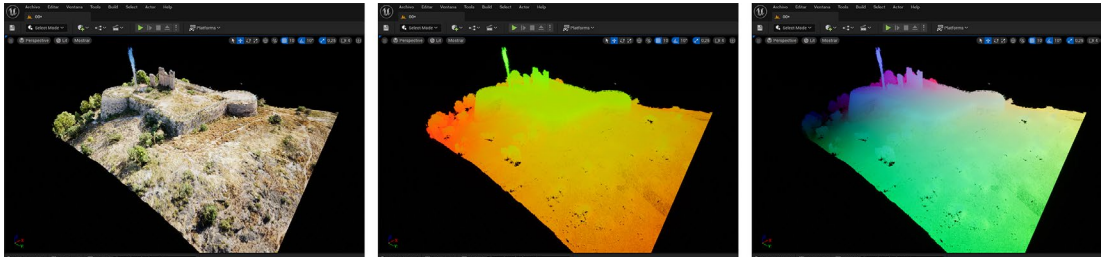


Fig. 7 - Colour mode: data, elevation and position.

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

with automatic LOD to improve performance so, it only loads in memory detailed things that are near the camera or the viewer, while loading a simplified version as things get further form the viewpoint. It also works in the same way in terms of the centre and corners of the screen respectively. This process, when it refers to rendering the points, represented as sprites, leads to a size distortion effect that affects quality negatively. The aforementioned variable sets the number of points that are compulsory rendered at a time in the screen, overriding the LOD effect and solving the issue with high values. Setting the variable between 3 to 6 million will grant a good representation (Fig. 8).

APPS DEVELOPMENT

Two apps will be developed in the research. The first of them, intended for more professional users, will be a sort of template UE project. It will contain all the main functionalities and assets needed for the visualization, except the cloud itself that will have to be imported by the user. The second one, aimed at a wider audience, will be a packaged executable able to run in any computer following certain simple instructions.

UE5 EDITOR POINT CLOUD VIEWER

The idea is to simplify as much as possible the work done in UE by the user so, this app contains all basic elements that the project needs for working properly [9]. As seen in Fig. 9, two level assets, the main asset that will contain the point cloud (BP_PC) and the asset of the background (BP_SPHERE) are placed in the root folder of the content browser, as well as four more folders. Here is where the point cloud has to be imported to then link it to the point cloud container (BP_PC) as shown in Fig. 10. Theoretically, there is no size limit for the point cloud, but if the import process crashes continuously or just does not start, it is advisable to decimate it. Clouds of 2,5 GB and around 100 million points will run smoothly in almost any standard computer [10].

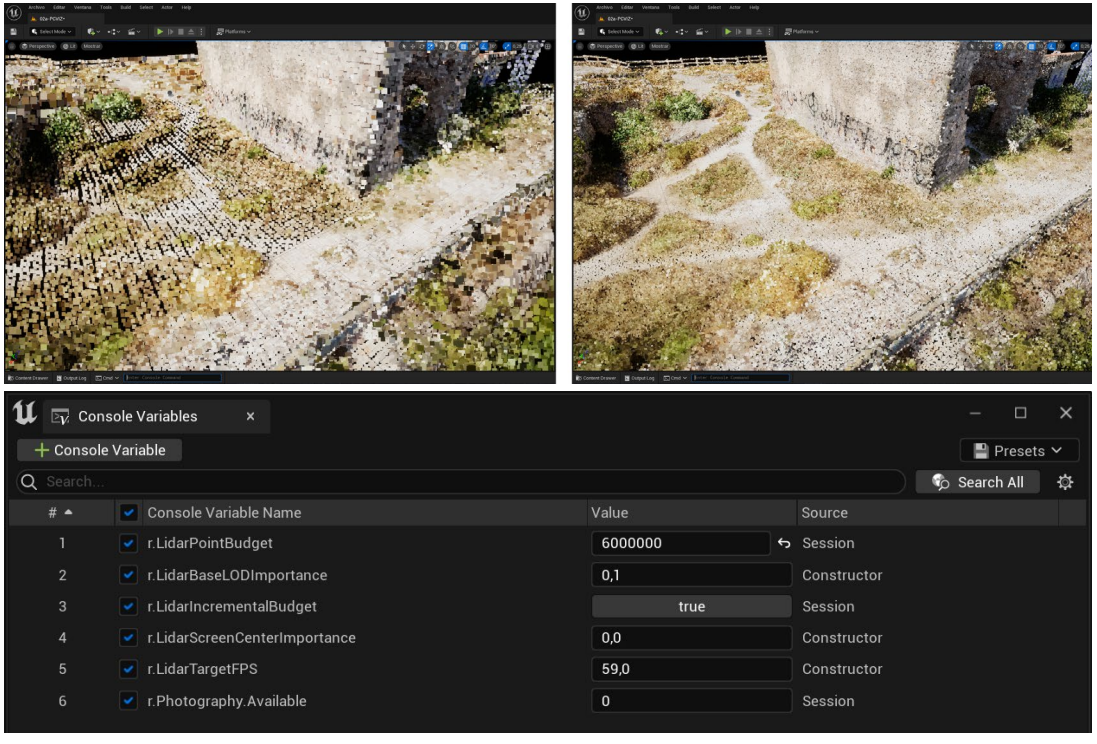


Fig. 8 - Console Variables. Point Budget comparison. Left: 1 mill points – Right: 6 mill points.

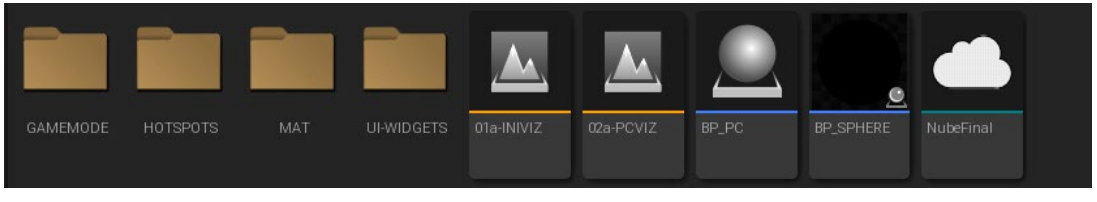


Fig. 9 - Content Browser in the Editor Point Cloud Viewer.

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

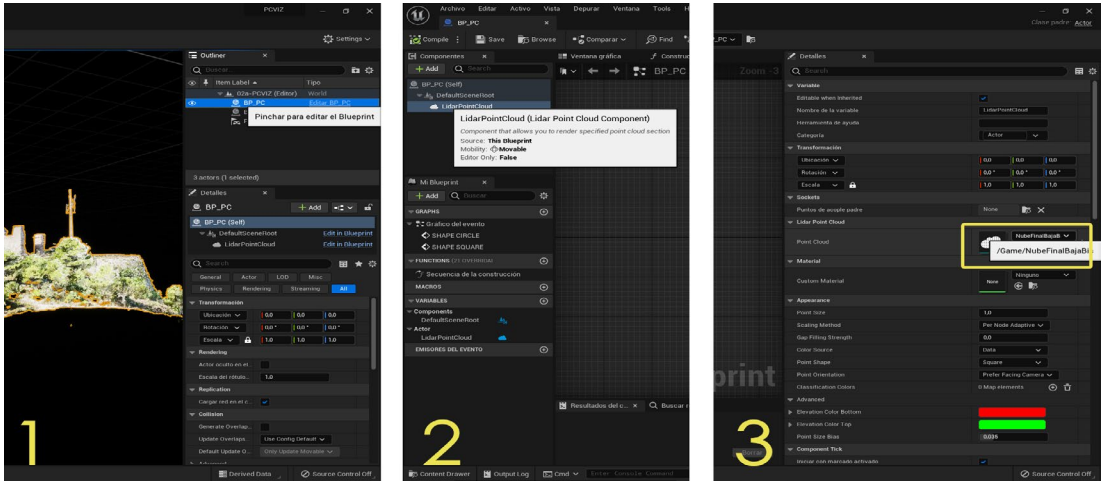


Fig. 10 - Linking the Point Cloud to the Blueprint object. 1 – open the asset, 2 – Select the LiDAR Point Cloud Component and 3 – In the side menu, select the LiDAR Point Cloud loaded in the project.

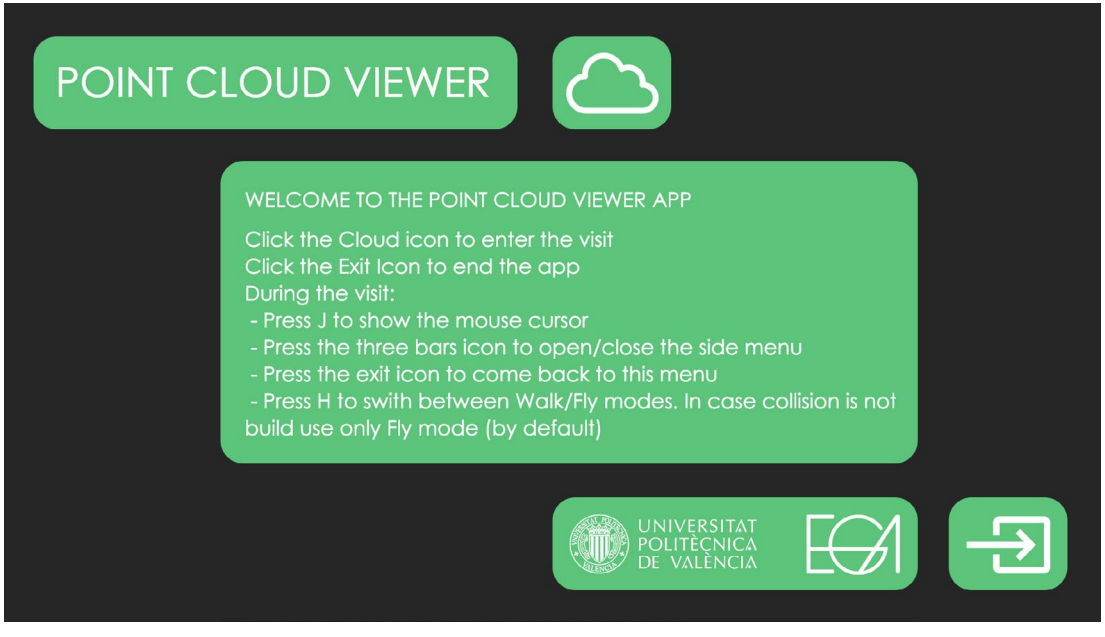


Fig. 11 - Main menu when starting the viewer.

Double clicking the point cloud gives access to the editor where collision can be calculated, although that is not compulsory so the visit will start always in Flying Mode. It is possible to change to Walking Mode if collision has been previously computed. As there is no prior size, composition or topography set for the point cloud, in order to start the visit in the desired location the asset Player Start has to be positioned properly. Specially in height when the collision is calculated because otherwise it will not be possible to pass through the points (Fig. 15). At this point the project is ready for packaging in shipping mode what will generate a folder with an executable file ready for the visit.

When starting the virtual visualization, the visitor will find a main menu giving access to the point cloud and the operating instructions for the visit (Fig. 11).

Pressing the "J" key on the keyboard will display the mouse cursor, which will be necessary to exit the visit or access to the side menu. This menu contains some buttons and sliders designed to modify the shape, size, gap and game mode of the cloud according to the user preferences and requirements (Fig. 12).

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

VISUALIZATION APP

Increasing the automation of processes and directly packaging the project developed for the previous app result in a greatly simple visualization tool. Exposing all assets, variables and functions to blueprint programming has allowed to create a very straightforward workflow in which the user only has to open the app, select the point cloud to load and enter directly into the visit, providing the same possibilities of changing the visual appearance of the cloud as in the previously exposed methodology.
To achieve this, all the same assets that existed in the preceding project have been modified internally through the use of blueprints. Now, when the point cloud loads, as it is linked to the container that already exists in the scene (BP_PC), it directly appears with default parameters that have been set. In this case two limitations apply. Firstly, the size of the file cannot be bigger than 1 GB. Secondly, the collision cannot be calculated and built in the asset of the point cloud. Consequently, the visualization can be only carried out in Fly Mode.
Independent programmers and videogame designers have asked Epic Games developers, many times, to implement collision cooking in runtime. This should not be impossible, especially when Unity 3D, the other software used massively in video game industry, has it fully functional [11]. Complex routines based in the addition of simple collision shapes adjusting to procedurally created meshes is one of the options that exist. But the real answer is that it is not easily possible to calculate collisions in runtime, particularly when the collision mesh has to be calculated from points using an algorithm that is not really packaged in the app itself and only runs in the editor mode.
In regard to the file size, UE works way better with a great number of small files than with one big file. It is possible to import several point clouds to put them all together in one scene in the engine during editing. Then it is needed to apply an alignment between them, so they fit perfectly into a unique model. But this action is not possible again in runtime and turns into the limitation indicated. Anyway, adjusting the shape and size parameters of the points gives good visualization results starting in clouds of 10 million points and 200 MB size that become astonishing when arriving to around 40 million points and 1 GB size. For some purposes the many points the better but not all components in a virtual environment need the highest detail level (Lindner et al., 2021) so a good balance between time requirements and performance is generally more desirable (Dickmann and Dunker, 2014).
In an extensive and intensive search, in different formats, only two or three people have talked in forums about runtime point cloud loadings. All of them were always talking about spawning the point cloud actor inside the scene according to the game script design but having it loaded in the project and cooked and packaged in the game. With the work developed in this research, a step forward in the management, visualization and divulgation of heritage is intended.

Fig. 12 - Interface during the visit with side menu opened.

THE WORKFLOW

As represented in Fig. 14, the workflow, as was initially intended, is made up with few steps. It contrasts with previous studies (Laksono et al., 2019) made for the same scope in which after the point cloud acquisition there are yet three software with complex operations to do prior to the visualization. 3D Fier for extrusion, QGIS for editing and conver-

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

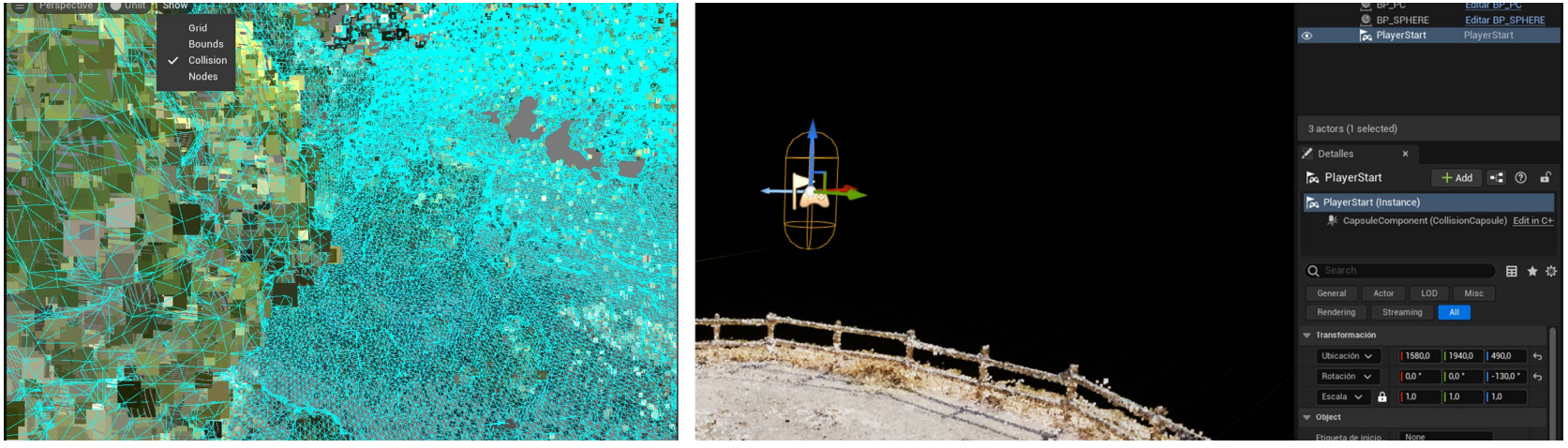


Fig. 13 - View of the collision mesh and the Player Start asset in the Editor.

sion, Mapbox for tiling and Unity 3D for the visualization, getting finally such a simplified model that is far from a real representation.
In other cases (Barszcz et al., 2023), (Carvajal et al., 2020) or (Wei O et al., 2019) procedures are simpler, but, as the acquisition is of small objects to place in a modelled virtual museum, the scale difference makes the comparison inconclusive. Moreover, when the main problem in point cloud management is their size and scalability which in these cases obviously disappears.
In a study to create a virtual tour of the Southern Campus of the Gebze Technical University (Turkey) (Sefercik et al., 2021) the workflow is laborious. A dense point cloud is generated with the data of an aerial and terrestrial survey and then studying its data, high-quality 3D textured mesh models were produced, and the environment completed with urban elements. But when it comes to heritage, the modelling of ruins, rests and environments is so difficult than is better to rely only on the point cloud in which all existing data is a truthful representation of reality. Every mesh compositing, procedural model or volume simplifications have an intrinsic interpretation of data in which some loss is produced. As Cipriani et al. (2019) reckon,

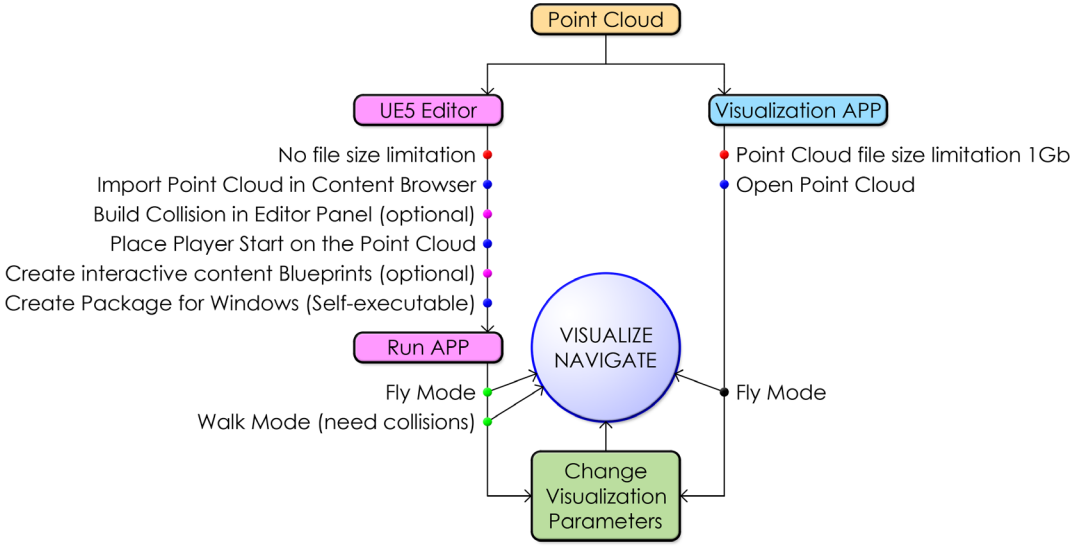


Fig. 14 - Workflow for the two apps.

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961


Fig. 15 - Comparison Cloud Point (left) – Model (right) in the SCG University. Source: Sefericik et al., 2021.

Fig. 16 - Screenshots during the visit to Santa Ana Castle (Oliva, Valencia) in Fly Mode

cultural heritage assets cannot be simplified losing the constructive and aesthetic qualities that have determined their "fortune". The superficial aspects of these artifacts are essential to allow the user, even virtual, to have a complete and conscious experience (Cipriani et al., 2019)

GENERATING AN INTERACTIVE VIRTUAL TOUR BASED ON POINT CLOUDS

The previously exposed methodology has been tested using a point cloud which was obtained from a survey of the Santa Ana Castle in the town of Oliva in Valencia (Spain). This monument was built in the 16th century to protect the town from the Berber pirates and nowadays is classified as a cultural interest asset. The castle is located on the hill of Santa Ana. It consists of a battered rectangular platform, measuring about 44 x 35 meters, which includes two circular shaped towers that are located at the ends of the diagonal of the plan, one at the north-west side part of the building, and the other at the south-east one. Over the platform still remain the ruins of an earlier Gothic hermitage containing only some walls since it was abandoned in the early 19th century (Soler, 2020). The only remaining inner structures of the fortress are the cistern for the water supply, which is located parallelly to the entrance ramp at the east side of the building and the space inside the south-east tower, which it is covered by an interesting thoric-shaped vault. There are no traces of any construction of the fortress over the platform, since King Felipe V of Spain ordered its demolition in 1708 (Blay, 1960). The lack of spaces is also striking at the interior of the northwest tower, which, despite of having embrasures in its walls like the southeast tower, seems to have been filled up with debris and ground (Rodríguez-Navarro, Gil & Ruggieri, 2023).
Figure 16 shows some screenshots from the interactive visit of the castle that has been carried out using the exposed methodology.
Going one more step further, a musealization visit providing graphical and written information was developed through the use of hotspots. The

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

project contains a Blueprint object (BP_HOTSPOTS-BASE) to create customizable instances in different points of the cloud. Once the asset is placed several times in the scene, each of them can be independently modified adding a text and an image. The hotspots appear during the visit in the shape of a red sphere that will show the information when clicked with the mouse (Fig. 17). In the example it can be seen the use of hotspots to show images and even a particular visit of an interior space through the use of a panorama (Fig. 18). The visualization of the panorama is achieved by making the visitor jump to a different level inside the game, appearing in the centre of a big sphere that has the equirectangular texture applied as a material. The player remains in a fixed position in this case to be always in the centre of projection thus perceiving the surrounding space properly. Watching panoramas is optional and is not intended to be included in the template project, but for advanced users of UE proves the strength of such a tool.
The template project for UE, the stand-alone app developed in this research and the interactive visit example of the Santa Ana Castle can be downloaded by following this link or by using the QR Code that is shown on the next page (Fig. 19)).

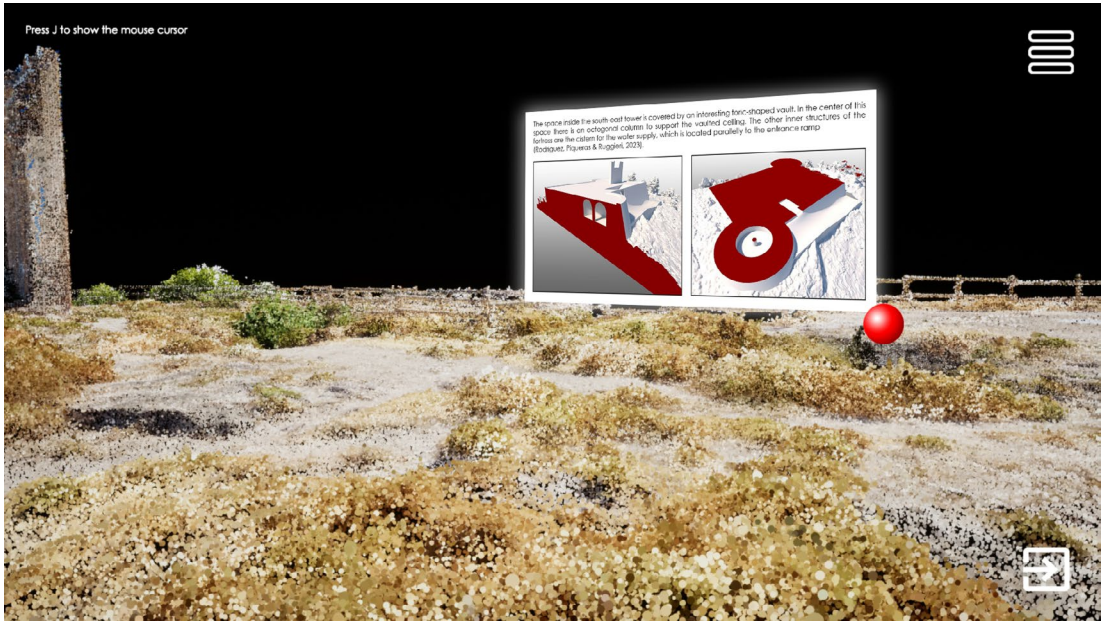


Fig. 17 - Interface during the visit with a hotspot showing its associated content as an image.

CONCLUSIONS

In conclusion, harnessing the power of a LiDAR point cloud viewer based in UE offers benefits and opportunities. The ability to visualize complex 3D data in real-time provides unprecedented insights. By leveraging the advanced rendering capabilities and interactivity of UE, users can navigate and explore the spaces acquired and represented through point clouds with ease, uncovering hidden patterns, details, and relationships. Especially when the walk through can be done at a person eye level. This empowers professionals

Fig. 18 - Viewing a spherical panorama of an inner space during the visit from a hotspot.

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

in fields such as urban planning, architecture and archaeology to gain a deeper understanding of their respective domains and to communicate it to a wider public.
The realism and fidelity offered by the app visual capabilities make it an ideal platform for showcasing data to both technical and non-technical audiences. However, it's important to note that while UE provides an exceptional platform for LiDAR point cloud visualization, there are considerations to keep in mind. These include ensuring the compatibility and optimization of the LiDAR data, managing large-scale point clouds efficiently, and addressing computational requirements for real-time rendering. As the technology continues to advance, we can expect further refinements and innovations, driving the utilization of the viewers in an even broader range of applications.

AKNOWLEDGEMENTS

This contribution is part of the R+D+i project PID2020-119469RB-I00, funded by the Ministry of Science and Innovation/ State Research Agency/ 10.13039/501100011033

Fig. 19 - QR code to download the template, the stand alone wiewer and the example

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

NOTES

[1] https://leica-geosystems.com/es-es/products/laser-scanners/software/leica-cyclone/leica-cyclone-3dr/leica-cyclone-3dr-download

[2] https://www.faro.com/es-MX/Products/3D-App-Center-Overview/Zone-Viewer-App

[3] https://www.unrealengine.com/en-US/eula/unreal
[4] https://dev.epicgames.com/community/

[5] https://docs.unrealengine.com/5.2/en-US/lidar-point-cloud-plugin-for-unreal-engine/

[6] A sprite is a 2D image or texture that can be rendered directly in a scene without being applied to any static mesh geometry.

[7] https://docs.unrealengine.com/5.2/en-US/lidar-point-cloud-plugin-overview-in-unreal-engine/

[8] For the specific definition of each of the variables go the plugin documentation (link in note 7).

[9] The template has been done with UE 5.0.3. Any newer version will open it without problems.

[10] Testing has been done in different desktop and laptop computers. With GPU of previous generations (GTX 1050 i) clouds until 2,5 GB run well on import and runtime.

[11] In any case is an option that has several limitations listed in Unity documentation. See https://docs.unity3d.com/Manual/class-MeshCollider.html

REFERENCES

Barszcz, M., Dziedzic, K., Skublewska-Paszkowska, M., & Powroznik, P. (2023) 3D Scanning Digital Models for Virtual Museums. *Computer Animation and Virtual Worlds*, vol. 34, no. 3-4, 2023, https://doi.org/10.1002/cav.2154

Blay Navarro, J. (1960). *Documentos y datos para la historia de la Ciudad de Oliva*. Valencia: ECIR.

Carvajal D., Morita M., & Bilmes G. (2020) Virtual museums. Captured reality and 3D modeling. *Journal of Cultural Heritage*. Vol. 45, 2020, pp. 234–243.

Cipriani, L., Bertacchi, S., & Bertacchi, G. (2019) An Optimised Workflow for the Interactive Experience with Cultural Heritage through Reality-Based 3d Models: Cases Study in Archaeological and Urban Complexes. *2ND INTERNATIONAL CONFERENCE OF GEOMATICS AND RESTORATION (GEORES 2019)*, vol. 42, no. 2, Copernicus Gesellschaft Mbh, 2019, pp. 427–34, https://doi.org/10.5194/isprs-Archives-XLII-2-W11-427-2019

Dickmann, F., & Dunker, S. (2014). Visualisierung von 3D-Gebäudemodellen - Welche Ansprüche stellt die Planung an dreidimensionale Stadtansichten. *Kartographische Nachrichten*, vol. 64, no. 1, 2014, pp. 10–16, Springer International Publishing.

Franczuk, J., Boguszewska, K., Parinello, S., Dell'Amico, A., Galasso, F., & Gleń, P. (2022). Direct use of point clouds in real-time interaction with the cultural heritage in pandemic and post-pandemic tourism on the case of Kłodzko Fortress. *Digital Applications in Archaeology and Cultural Heritage*, 24, e00217. https://doi.org/10.1016/j.daach.2022.e00217

Kumar, A. (2020). *VR Integrated Heritage Recreation: Using Blender and UE 4*. Berkeley: Apress. https://doi.org/10.1007/978-1-4842-6077-7

Laksono, D., Aditya, T., & Riyadi, G. (2019) Interactive 3d city visualization from structure motion data using game engine. I*nternational Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 42, no. 4, 2019, pp. 737–40, https://doi.org/10.5194/isprs-archives-XLII-4-W16-737-2019

Lindner, C., Ortwein, A., Staar, K., & Rienow, A. (2021). Different Levels of Complexity for Integrating Textured Extra-terrestrial Elevation Data in Game Engines for Educational Augmented and Virtual Reality Applications. *KN - Journal of Cartography and Geographic Information,* vol. 71, no. 4, 2021, pp. 253–67, Springer International Publishing. https://doi.org/10.1007/s42489-021-00090-3

Rodríguez-Navarro, P., Gil Piqueras, T., & Ruggieri, A. (2023). Levantamiento gráfico integral para el análisis de la Fortaleza de Santa Ana en Oliva (Valencia). In M. G. Bevilacqua & D. Ulivieri (Eds.), *Defensive architecture of the mediterranean,* vol. XV (pp. 1131-1138). Pisa: Pisa University Press. http://digital.casalini.it/10.12871/9788833397948142

Sefercik, U. G., Kavzoglu, T., Nazar, M., Atalay, C., & Madak, M. (2021) UAV-BASED 3D VIRTUAL TOUR CREATION. International Archives of the Photogrammetry, *Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 46, no. 4, Copernicus GmbH, 2021, pp. 493–99, https://doi.org/10.5194/isprs-Archives-XLVI-4-W5-2021-493-2021

Soler Molina, A. (2020). *L'esplendor d'Oliva. De l'honor dels Carròs al comtat dels Centelles*. Oliva: Ajuntament de Oliva.

Wei O., Majid Z., & Setan H. (2019). Three-dimensional recording and photorealistic model reconstruction for virtual museum application–an experience in Malaysia. *8th International Workshop 3d-Arch: 3d Virtual Reconstruction And Visualization Of Complex Architectures*, vol. 42, no. 2, Copernicus Gesellschaft Mbh, 2019, pp. 763–71, https://doi.org/10.5194/isprs-archives-XLII-2-W9-763-2019

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

Visor universal de nubes de puntos basado en el motor de videojuegos Unreal Engine

Las nubes de puntos son una herramienta esencial para documentar el patrimonio arqueológico y arquitectónico, dado que son el resultado primigenio fruto de un escaneado láser o de un levantamiento fotogramétrico. A pesar de ser una fuente de información muy básica y discretizada, que contiene únicamente puntos coloreados (Fig. 1), puede utilizarse como una primera aproximación al modelo mediante una visualización virtual o recorrido interactivo. La gestión y visualización de nubes de puntos no es una tarea sencilla, ya que implica la representación de millones de puntos y es necesaria la instalación de visores específicos. La mayoría de estos visores especiales son programas comerciales, pero existen algunas alternativas gratuitas. Casi todos los fabricantes de escáneres láser proporcionan su propio visor gratuito. Por ejemplo, Leica ofrece el *software* Leica Cyclone 3DR Viewer [1]. Faro también suministra gratuitamente su *software* Faro Zone Viewer [2], pero sólo es capaz de visualizar algunos formatos propietarios de los productos desarrollados por la firma. Los motores de videojuegos como Unreal Engine (UE) son excelentes entornos de visualización por estar optimizados para aprovechar al máximo los recursos del *hardware*. Además de esto, UE es gratuito para fines no comerciales y para proyectos con ganancias inferiores a 1 millón de dólares [3]. Permite crear elementos que interactúan con el espectador, gracias a su entorno de programación gráfica basado en *blueprint*s o incluso utilizando el lenguaje de programación tradicional C++. De este modo, es posible incluir información interactiva adicional que pueda usarse para generar un recorrido virtual musealizado de la nube de puntos. Otros investigadores como Franczuk et al. (2022) y Kumar (2020) han podido evaluar también el potencial de UE aplicado a la visualización de nubes de puntos del patrimonio cultural, pero este artículo se centrará en desarrollar un flujo de trabajo y una metodología adecuada para crear y compartir módulos de visualización que permitirán la navegación de nubes de puntos de una manera sencilla y universal.

GESTIÓN DE NUBES DE PUNTOS CON UE

UE es un *software* de Epic Games. Fue diseñado inicialmente con el único propósito de crear juegos en primera persona, pero, debido a su potencial, se utilizó también para el desarrollo de otras aplicaciones (Kontos, 2020). Aunque está diseñado para iluminar, sombrear, animar, componer y organizar entornos virtuales, carece de herramientas de modelado avanzado. Esto significa que la mayoría de los elementos físicos deben importarse y colocarse en la escena. Luego deben aplicarse materiales, luces, animaciones, sistemas de partículas, colisiones y muchas otras características que completan un mundo virtual dentro de UE. Trabajar con nubes de puntos sigue el mismo flujo de trabajo, pero no se pueden importar directamente puesto que es necesario utilizar el com-

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

plemento o *plugin* denominado LiDAR *Point Cloud* para gestionarlas. Este *plugin* fue desarrollado por un programador anónimo independiente y se publicó en el bazar de Epic Games, de forma completamente gratuita, en febrero de 2018, tras concederle la beca Unreal Dev Grant a su autor. En mayo de 2020, después de siete versiones, finalmente se incorporó al editor, aunque es necesario activarlo específicamente. Su desarrollador, muy activo en la *Epic Games Dev Community* [4], llevó a cabo una mejora continua del *plugin* en base a los comentarios recibidos en el foro, pero dejó de responder solicitudes desde febrero de 2022. El proyecto parece estar estancado, por el momento, en la versión 0.8 y sigue igual incluso con el gran cambio que UE ha realizado al pasar de la versión 4 a la 5. El *plugin* es todavía una versión beta, bastante estable para el uso que se pretende. En la documentación del *plugin* [5] se especifica que cuenta con las siguientes características: Soporta la mayoría de los formatos de archivo para nubes de puntos, archivos ASCII (TXT, XYZ, PTS) y LAS, e57; admite sombras dinámicas, tanto proyectadas como recibidas; representa los puntos como píxeles o como *sprites* [6]; implementa una técnica de iluminación tipo cúpula para ensalzar los volúmenes; tiene soporte para *blueprint*s; permite múltiples técnicas de coloreado (RGB, Intensidad, Elevación, Clasificación); tiene un sistema LOD (*Level Of Detail*) dinámico y GPU *streaming* que permite la carga de grandes archivos y, finalmente, permite la importación y modificación de datos asíncronos durante el tiempo de ejecución (Fig. 2). Todas estas características constituyen un potente conjunto de herramientas para trabajar con nubes de puntos y pueden simplificar la gestión de grandes conjuntos de datos en lo que respecta a la visualización. Además de ello, UE5 es capaz de representar los detalles más pequeños de un modelo 3D, incluso con un número muy alto de entidades, con un impacto insignificante en el rendimiento. Por tanto, los principales objetivos de esta investigación se centrarán en desarrollar una metodología que permita crear y publicar visitas virtuales que puedan ser también interactivas, centrándose especialmente en la posibilidad que ofrece UE para crear módulos autónomos o autoejecutables que puedan abrirse en cualquier ordenador, sin instalar ningún *software* específico.

Otro resultado obtenido de no depender del uso de UE será proporcionar una aplicación independiente completamente funcional (construida con UE) que puede cargar nubes de puntos y puede actuar como una herramienta de visualización potente y útil, simplificando mucho el proceso a costa de algunas limitaciones, como se mencionará más adelante.

EL *PLUGIN* DE NUBES DE PUNTOS LiDAR

Como se explicó anteriormente, al crear un nuevo proyecto en UE, es necesario activar el *plugin* de nube de puntos LiDAR, en el panel de complementos, para poder importar nubes de puntos. Al ser una versión beta da una advertencia que hay que tener en cuenta, pero después de todo su uso durante esta investigación se puede considerar que el *plugin* es estable en todo lo relacionado con la visualización. Por el contrario, cuando se trata de editar, modificar o exportar el rendimiento se vuelve inestable.
En el panel configuración del proyecto (Fig. 3a), es posible cambiar el *octree* y los parámetros de rendimiento, pero los valores están tan optimizados que esto no es necesario. Para cada valor físico, UE trabaja en centímetros, por lo que existe un valor de escala de importación/exportación que transforma las unidades correctamente. Finalmente, el apartado de automatización tiene dos parámetros importantes; el primero es el "centrado automático al importar", marcado de forma predeterminada porque preservar las coordenadas originales puede causar una pérdida notable de precisión si los valores son demasiado grandes; el segundo es la "creación automática de colisión al importar" que puede ser interesante en nubes de puntos ligeras, pero que puede fallar al importar nubes pesadas. En esos casos, es mejor crear la colisión en el panel de edición del objeto (Fig. 3b).
Al importar archivos de nubes de puntos grandes, el motor requiere una cantidad significativa de RAM para cargar datos y convertir la nube en un objeto de UE. Una vez que el archivo está completamente cargado, el *software* mantendrá el objeto en la memoria del sistema y solo cargará una cierta cantidad de puntos según sea necesario, según el recuento de LOD. Esto da como resultado una reducción en el consumo de RAM y, en consecuencia, una carga rápida de objetos. En la descripción general del *plugin* [7], podemos encontrar los indicadores de rendimiento sobre la carga de datos LiDAR, disponibles en abierto para el público, de la ciudad de Montreal (Fig. 4), pero estos son siempre valores ideales logrados con computadoras de muy alta gama y durante el estudio se comprueban valores más realistas. También hay que señalar que en el ejemplo todo el modelo está dividido en 621 archivos de aproximadamente 370 MB cada uno, lo que no siempre es la situación habitual. Es bastante más habitual disponer de un archivo único de gran tamaño, principalmente en el estudio de patrimonio donde generalmente se necesita una composición global para depurar y organizar la información.
Una vez que la nube de puntos está en la escena, se pueden leer y modificar veinte parámetros diferentes, pero el foco estará, en nuestro caso, en aquellos que afectan a la percepción. Como los puntos se representan como *sprites*, su forma se puede cambiar y escalar según sea necesario. Las formas de círculo y cuadrado son las predeterminadas (Fig. 5), pero creando un material de *sprite* es posible adoptar cualquier forma (Fig. 6b). Cuando el tamaño del punto se establece en cero, los puntos se representan como píxeles (Fig. 6a) y, en general, las formas cuadradas llenan mejor los espacios entre puntos pero dan una sensación más plana. Otros parámetros, como el método de escalado, la fuerza de relleno de espacios y la orientación del punto, se pueden modificar con menos afectación en la imagen final. La fuente del color permitirá elegir el canal de información de color entre datos, elevación, posición y clasificación (Fig. 7). Si no se elige ningún canal, entonces el color será el dado por el material aplicado a la nube de puntos.
En cuanto a la iluminación y las sombras, que tienen un gran impacto en la percepción de calidad de la representación, el flujo de trabajo es sencillo. Como las nubes generalmente vienen con el color real capturado en el escaneo, no es necesario colocar actores de luces. Esto también evita la necesidad de construir iluminación y, por tanto, elimina los procesos de cálculo. Para poder ver las nubes con el color y la luz con la que fueron escaneadas, el complemento crea el material de

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

los *sprites* automáticamente con la información de color aplicada en los canales base y emisivo.
Varias variables de consola relacionadas con las nubes de puntos se crean cuando se activa el *plugin* [8]. Sus valores por defecto están optimizados para usos generales y su influencia en la percepción es mínima a excepción de la variable *r.LidarPointBudget*. UE5 funciona con LOD automático para mejorar el rendimiento, por lo que solo carga en la memoria elementos detallados que están cerca de la cámara o del espectador, mientras que utiliza una versión simplificada a medida que los objetos se alejan del punto de vista. También funciona de la misma forma en cuanto al centro y esquinas de la pantalla respectivamente. Este proceso, cuando se renderizan los puntos de la nube como *sprites*, produce un efecto de distorsión de tamaño que afecta negativamente a la calidad. La variable antes mencionada establece la cantidad de puntos que se representan a la vez en la pantalla, anulando el efecto LOD y resolviendo el problema con valores altos. Cuando se establecer la variable entre 3 y 6 millones, se obtiene una óptima representación de los puntos (Fig. 8).

DESARROLLO DE APLICACIONES

En la investigación se desarrollarán dos modalidades. La primera de ellas, destinada a usuarios más profesionales, será crear una especie de plantilla del proyecto de UE. Esta plantilla contendrá todas las funcionalidades y elementos necesarios para la visualización, exceptuando la propia nube, que deberá ser importada por el usuario. La segunda modalidad, dirigida a un público más amplio, se basa en aportar un archivo ejecutable autónomo capaz de abrirse en cualquier ordenador siguiendo unas sencillas instrucciones.

VISOR DE NUBES DE PUNTOS CON EL EDITOR DE UE5

La idea es simplificar al máximo el trabajo que deba realizar el usuario en UE, por lo que esta aplicación contiene todos los elementos básicos que el proyecto necesita para funcionar correctamente [9]. Tal como se aprecia en la Fig. 9, en la carpeta raíz del *Content Browser* existen: dos niveles, el objeto principal que contiene la nube de puntos (BP_PC) y el objeto del entorno (BP_SPHERE), así como cuatro carpetas más. Aquí es donde se debe importar la nube de puntos para luego vincularla al contenedor de nubes de puntos (BP_PC) como se muestra en la Fig. 10. Teóricamente, no hay límite de tamaño para la nube de puntos, pero si el proceso de importación falla o no arranca, es aconsejable decimar la nube para reducirla. Las nubes de 2,5 GB y alrededor de 100 millones de puntos funcionarán sin problemas en casi cualquier ordenador estándar [10].
Al hacer doble clic en la nube de puntos, se accede al editor donde se pueden calcular las colisiones, aunque no es obligatorio. La visita comenzará siempre en Modo Vuelo, pero es posible cambiar al modo Caminar, si previamente se ha calculado la colisión. Como no hay un tamaño, composición o topografía establecidos previamente para la nube de puntos, para comenzar la visita en la ubicación deseada, el objeto *Player Start* debe estar posicionado correctamente. Debe prestarse especial atención a la altura del *Player Start*, cuando se calcula la colisión, porque de lo contrario no será posible andar sobre la nube (Fig. 15). Llegados a este punto, el proyecto está listo para compilarse, lo que generará una carpeta con un archivo ejecutable listo para realizar el recorrido por la nube.
Al iniciar la visualización virtual, el visitante encontrará un menú principal dando acceso a la nube de puntos y con las instrucciones de funcionamiento de la visita (Fig. 11). Al pulsar la tecla "J" del teclado, se mostrará el cursor del ratón, que será necesario para salir de la visita o acceder al menú lateral. Este menú contiene algunos botones y controles deslizantes diseñados para modificar la forma, tamaño, espacio y modo de movimiento sobre la nube, según las preferencias y requisitos del usuario (Fig. 12).

APLICACIÓN DE VISUALIZACIÓN

Automatizando los procesos y compilando directamente el proyecto de la aplicación anterior da como resultado una herramienta de visualización enormemente sencilla. Los objetos, variables y funciones proporcionados por la programación basada en *blueprints* permiten crear un flujo de trabajo muy sencillo en el que el usuario sólo tiene que abrir la aplicación, seleccionar la nube de puntos a cargar y adentrarse directamente en la visita, ofreciendo las mismas posibilidades de configuración ofrecidas por la metodología expuesta anteriormente.
Para lograr esto, todos los objetos que existían en el proyecto anterior se han modificado internamente mediante el uso de *blueprints*. De este modo, cuando se carga la nube de puntos, al estar vinculada al contenedor existente en la escena (BP_PC), aparece directamente con los parámetros predeterminados que se han configurado. En este caso se aplican dos limitaciones. En primer lugar, el tamaño del archivo no puede ser superior a 1 GB. En segundo lugar, la colisión no se puede calcular. En consecuencia, la visualización sólo se puede realizar en Modo Vuelo. Muchos programadores independientes y diseñadores de videojuegos han pedido a los desarrolladores de Epic Games, en numerosas ocasiones, que implementen el cálculo de colisiones en tiempo de ejecución. Esto no debería ser imposible, especialmente cuando Unity 3D, el otro *software* utilizado masivamente en la industria de los videojuegos, lo tiene completamente funcional [11]. Una de las opciones que existen son rutinas complejas basadas en la adición de formas de colisión simples que se ajustan a mallas creadas procedimentalmente. Pero la verdadera respuesta es que no es sencillo calcular colisiones durante la ejecución, particularmente cuando la malla de colisión debe calcularse a partir de puntos utilizando un algoritmo que en realidad no está empaquetado en la aplicación misma y solo se ejecuta en el modo de editor.
En cuanto al tamaño del archivo, UE funciona mucho mejor con una gran cantidad de archivos pequeños que con un archivo grande. Es posible importar varias nubes de puntos para juntarlas todas en una escena en el motor durante la edición. Luego es necesario aplicar una alineación entre ellas, para que encajen perfectamente en un modelo único. Pero esta acción no es posible nuevamente en tiempo de ejecución y se convierte en la limitación indicada. De todos modos, ajustando los parámetros de forma y tamaño de los puntos se obtienen buenos resultados de visualización a partir de nubes de 10 millones de puntos y 200 MB de tamaño que se vuelven sorprendentes al llegar a alrededor de 40 millones de puntos y 1

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

GB de tamaño. Para algunos propósitos, cuantos más puntos sean, mejor, pero no todos los componentes de un entorno virtual necesitan el mayor nivel de detalle (Lindner et al., 2021), por lo que generalmente es más deseable un buen equilibrio entre los requisitos de tiempo y el rendimiento (Dickmann y Dunker, 2014).
En una búsqueda extensa e intensiva, en diferentes formatos, sólo dos o tres personas han hablado en foros sobre cargas de nubes de puntos en tiempo de ejecución. Todos ellos siempre hablaban de insertar la nube de puntos dentro de la escena de acuerdo con el diseño del guion de juego, pero teniéndola cargada en el proyecto y compilada y empaquetada en el juego. Con el trabajo desarrollado en esta investigación se pretende dar un paso adelante en la gestión, visualización y divulgación del patrimonio.

EL FLUJO DE TRABAJO

Como se representa en la Fig. 14, el flujo de trabajo, tal como se pretendía inicialmente, se compone de unos pocos pasos. Contrasta con estudios previos (Laksono et al., 2019) realizados para el mismo ámbito en los que tras la adquisición de la nube de puntos aún quedan tres programas que usar, con operaciones complejas por realizar, antes de la visualización. 3D Fier para la extrusión, QGIS para la edición y conversión, Mapbox para el mosaico y Unity 3D para la visualización, consiguiendo finalmente un modelo tan simplificado que dista mucho de una representación real.
En otros casos (Barszcz et al., 2023), (Carvajal et al., 2020) o (Wei O et al., 2019) los procedimientos son más sencillos, pero, como la adquisición es de pequeños objetos para colocar en un museo virtual modelado, la diferencia de escala hace que la comparación no sea concluyente. Aún más, cuando el tamaño y escalabilidad, como principales problemas en la gestión de nubes de puntos, en estos casos evidentemente desaparece.
En un estudio para crear un recorrido virtual por el Campus Sur de la Universidad Técnica de Gebze (Turquía) (Sefercik et al., 2021) el flujo de trabajo es laborioso. Se genera una densa nube de puntos con los datos de un estudio aéreo y terrestre y luego, estudiando sus datos, se producen modelos de malla texturizada 3D de alta calidad y se completa el entorno con elementos urbanos. Pero cuando se trata de patrimonio, la modelización de ruinas, restos y entornos es tan difícil que es mejor confiar únicamente en la nube de puntos en la que todos los datos existentes son una representación veraz de la realidad. Toda composición de mallas, modelo procedural o simplificación de volúmenes tiene una interpretación intrínseca de los datos en la que se produce alguna pérdida. Como Cipriani et al. (2019) opinan, los bienes de patrimonio cultural no pueden simplificarse perdiendo las cualidades constructivas y estéticas que han determinado su "fortuna". Los aspectos superficiales de estos elementos son esenciales para permitir al usuario, incluso virtual, tener una experiencia completa y consciente (Cipriani et al., 2019).

GENERANDO UN RECORRIDO VIRTUAL INTERACTIVO BASADO EN NUBES DE PUNTOS

La metodología expuesta anteriormente se ha probado utilizando una nube de puntos que se obtuvo a partir de un levantamiento del Castillo de Santa Ana en la localidad de Oliva en Valencia (España). Este monumento fue construido en el siglo XVI para proteger la localidad de los piratas berberiscos y hoy en día está catalogado como bien de interés cultural. El castillo está situado en el cerro de Santa Ana. Consta de una plataforma rectangular ataluzada, de unos 44 x 35 metros, que incluye dos torres de forma circular que se ubican en los extremos de la diagonal de planta, una en la parte lateral noroeste del edificio, y la otra la sureste. Sobre la plataforma aún se conservan las ruinas de una ermita gótica anterior que contiene sólo algunos muros desde que fue abandonada a principios del siglo XIX (Soler, 2020). Las únicas estructuras interiores que quedan de la fortaleza son la cisterna para el suministro de agua, que se encuentra paralela a la rampa de entrada en el lado este del edificio y el espacio interior de la torre sureste, que está cubierta por una interesante bóveda de forma tórica. No quedan restos de construcción alguna de la fortaleza sobre la plataforma, ya que el rey Felipe V de España ordenó su demolición en 1708 (Blay, 1960). Llama la atención también la falta de espacios en el interior de la torre noroeste, que, a pesar de tener troneras en sus muros como la torre sureste, parece haber sido rellena de escombros y tierra (Rodríguez-Navarro, Gil & Ruggieri, 2023).
La Figura 16 muestra algunas capturas de pantalla de la visita interactiva al castillo que se ha realizado mediante la metodología expuesta.
Yendo un paso más allá, se desarrolló una visita de musealización que proporciona información gráfica y escrita mediante el uso de *hotspots*. El proyecto contiene un objeto *Blueprint* (BP_HOTSPOTS-BASE) para crear instancias personalizables en diferentes puntos de la nube. Una vez colocado el objeto varias veces en la escena, cada una de ellas se puede modificar de forma independiente añadiendo un texto y una imagen. Los *hotspots* aparecen durante la visita en forma de una esfera roja que mostrará la información al hacer clic con el ratón (Fig. 17). En el ejemplo se puede observar el uso de los mismos para mostrar imágenes e incluso una visita particular de un espacio interior mediante el uso de una panorámica (Fig. 18).
La visualización del panorama se logra haciendo que el visitante salte a un nivel diferente dentro del juego, apareciendo en el centro de una gran esfera que tiene la textura equirectangular aplicada como material. El jugador permanece en una posición fija en este caso para estar siempre en el centro de proyección percibiendo así correctamente el espacio circundante. Ver panoramas es opcional y no está destinado a incluirse en el proyecto de plantilla, pero para los usuarios avanzados de UE demuestra la solidez de dicha herramienta.
El proyecto de plantilla para UE, la aplicación *stand-alone* desarrollada en esta investigación y el ejemplo de visita interactiva del Castillo de Santa Ana, pueden descargarse siguiendo este enlace o utilizando el Código QR que se muestra a continuación.

CONCLUSIONES

Aprovechar el potencial de un visor de nubes de puntos LiDAR basado en UE ofrece muchas ventajas. La capacidad de visualizar datos 3D complejos en tiempo real proporciona una fluidez sin precedentes. Al aprovechar las capacidades avanzadas de renderizado y la interactividad de UE, los usuarios pueden navegar y explorar los espacios

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

adquiridos y representados a través de nubes de puntos con facilidad. Especialmente cuando el recorrido se puede realizar a la altura de los ojos de una persona. Esto permite a los profesionales de campos como el urbanismo, la arquitectura y la arqueología obtener una comprensión más profunda de sus respectivos dominios y comunicarlos a un público más amplio. El realismo y la fidelidad que ofrecen las capacidades visuales de la aplicación la convierten en una plataforma ideal para mostrar datos tanto a audiencias especializadas como al público general. Sin embargo, es importante tener en cuenta que, si bien UE proporciona una plataforma excepcional para la visualización de nubes de puntos LiDAR, hay consideraciones que tener en cuenta. Ello implica garantizar la compatibilidad y optimización de los datos LiDAR, gestionar eficientemente las nubes de puntos a gran escala y abordar los requisitos computacionales para la representación en tiempo real. A medida que la tecnología continúa avanzando, podemos esperar mejoras e innovaciones, impulsando la utilización de visores a una gama aún más amplia de aplicaciones.

AGRADECIMIENTOS

Esta contribución es parte del proyecto de I+D+i PID2020-119469RB-I00, financiado por el Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación/ 10.13039/501100011033.

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13

ISSN 1828-5961

NOTAS

[1] https://leica-geosystems.com/es-es/products/laser-scanners/software/leica-cyclone/leica-cyclone-3dr/leica-cyclone-3dr-download

[2] https://www.faro.com/es-MX/Products/3D-App-Center-Overview/Zone-Viewer-App [3] https://www.unrealengine.com/en-US/eula/unreal

[4] https://dev.epicgames.com/community/

[5] https://docs.unrealengine.com/5.2/en-US/lidar-point-cloud-plugin-for-unreal-engine/

[6] Un sprite es una imagen o textura 2D que se puede representar directamente en una escena.

[7] https://docs.unrealengine.com/5.2/en-US/lidar-point-cloud-plugin-overview-in-unreal-engine/

[8] Para la definición específica de cada una de las variables, consulte la documentación del plugin (enlace en la nota 7).

[9] La plantilla se ha realizado con UE 5.0.3. Cualquier versión más reciente la abrirá sin problemas.

[10] Se han realizado pruebas en diferentes ordenadores de sobremesa y portátiles. Con GPUs de generaciones anteriores (GTX 1050 i), las nubes de hasta 2,5 GB funcionan bien en importación y tiempo de ejecución.

[11] En cualquier caso es una opción que tiene varias limitaciones enumeradas en la documentación de Unity. Consulte https://docs.unity3d.com/Manual/class-MeshCollider.html

DOI: https://doi.org/10.20365/disegnarecon.30.2023.13