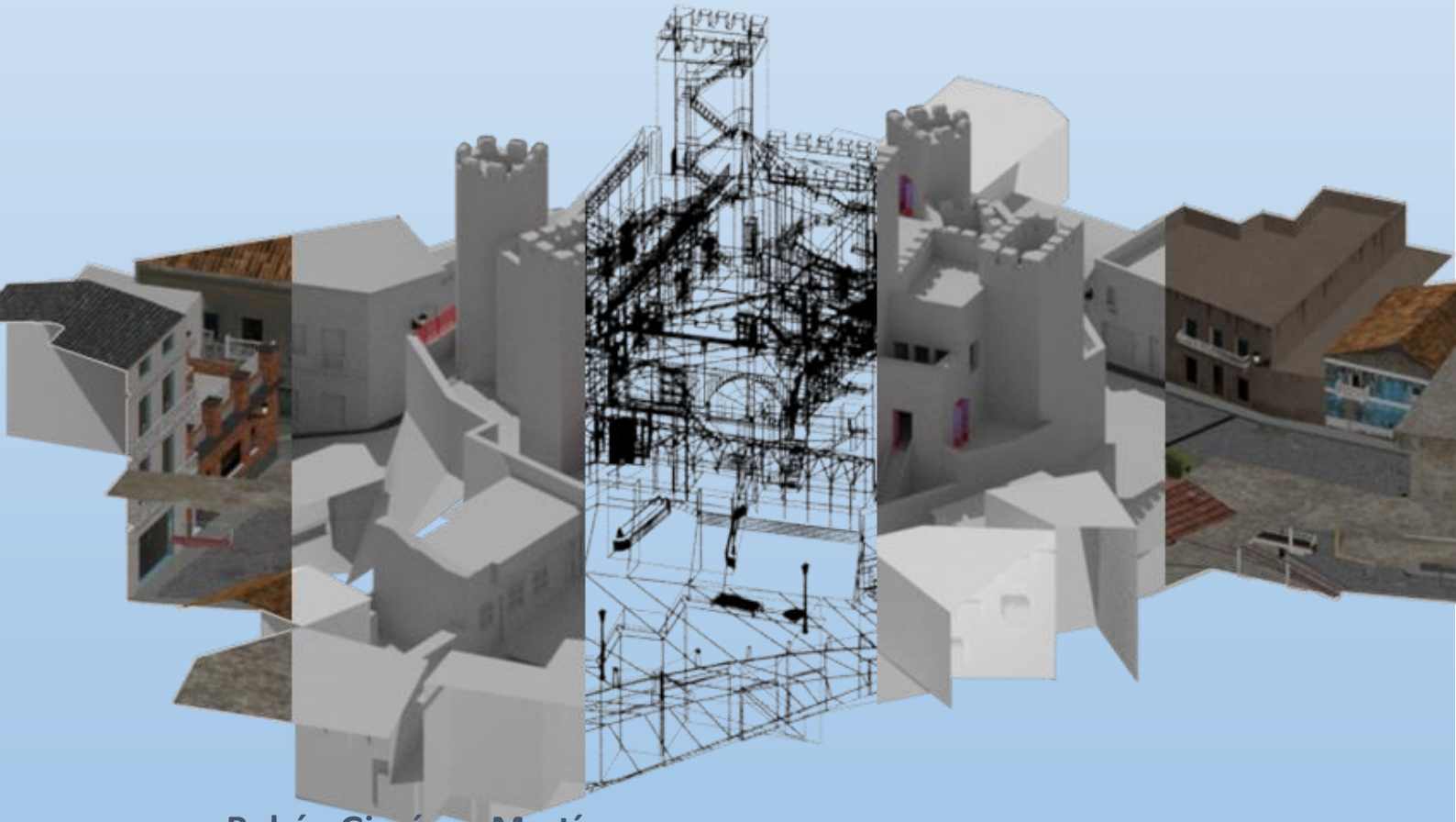
Escaneado láser 3D y renderizado multiplataforma del Castillo de Bétera

Proyecto de levantamiento láser escáner terrestre del castillo de Bétera, modelado, texturizado y exportación a motor gráfico de renderizado en tiempo real.

Trabajo fin de grado – Graduado en Ingeniería Geomática y Topografía



Rubén Giménez Martínez

Tutor: José Luis Lerma García

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR

1. Índice

Agradecimientos

Me gustaría dar mi agradecimiento a todas aquellas personas que han hecho posible la elavoración de este Trabajo Fin de Grado.

En primer lugar agradecer al tutor de este proyecto, José Luis Lerma García, por todo el apoyo, ideas, sugerencias y confianza que a depositado tanto en le proyecto, como en mi persona. Siempre he tenido sus puertas abiertas para cualquier consulta o problemática que haya podido surgir, ademas de guiarme en todo cuanto he necesitado.

No menos importante el agradecimiento a todo el personal docente de la facultad, por brindarme todos aquellos conocimientos e ideas que he ido adquiriendo en los sucesivos cursos de grado.

A Toni García, topógrafo del ayuntamiento de Bétera, por su implicación e interes en el proyecto, ademas de abrirme las abrirme las puertas de cuanto necesitara.

A todo aquel personal del ayuntamiendo de Bétera que me ha facilitado toda la información y acceso de cuanto he necesitado.

Tambien agradecer a todos aquellos compañeros de fatigas con los que he podido compartir la experiencia y placer de estudiar este grado.

Finalmente a mis padres, por confiar en mi y aguantarme todos estos años.

2. Introducción y objetivo del proyecto

El presente proyecto pretende abordar la creación de un entorno virtual interactivo del castillo de Bétera mediante la combinación de datos obtenidos mediante un escaneado 3D del patio interior del castillo, planos proporcionados por el Ayuntamiento de Bétera y fotografías del entorno exterior del castillo.

La combinación de estos tres elementos ha permitido la realización de un modelo completo de los exteriores del catillo, la fachada y su estructura interna.

Para concluir este modelo se importaría a un motor de renderizado a tiempo real para que el usuario que desee visualizarlo pueda ver y conocer el castillo, permitiendo un libre movimiento por todo el entorno generado (Fig. 2.1).

El proyecto es compatible con navegadores compatibles con HTML5, PC y dispositivos Android.

La fluidez del mismo estará limitada por las características técnicas del propio dispositivo, aunque se ha tenido en cuenta la mayor optimización del mismo para que resulte ser fluido en la mayoría de dispositivos actuales.

Figura 2.1. Castillo de Bétera Unity.

3. Localización

El término de Bétera, con una superficie de 75,67 km^2 pertenece a la comarca del Camp de Túria, situada en la provincia de Valencia, a una altitud de 120 m y a una distancia de 18 km de su capital y a 23 km del mar (Fig. 3.1).

El municipio tiene, en la actualidad 22.249 habitantes (Datos del Padrón de Habitantes a Fecha: 2015). Consta de un núcleo urbano central y de varios núcleos de población en el resto del término municipal, con varias urbanizaciones en cada uno de ellos.

Se sitúa en las últimas estribaciones de la vertiente sur de la Sierra Calderona, limitando con la comarca de -L' Horta de Valéncia.

Sus límites son, al norte, las poblaciones de Náquera y Serra; al este, la de Moncada; al oeste, las de La Pobla de Vallbona, San Antonio de Benagéber y L' Eliana; al sur, las de Paterna, Godella, Moncada y Valencia (Fig. 3.2).

Comunicada con la capital a través de las carreteras de Valencia-Burjassot-Torres Torres, la de Valencia-Ademuz y otras municipales, así como con los ferrocarriles de la Generalitat Valenciana, que permiten el acceso al centro mismo de Valencia.

Figura 3.1. Localización Bétera.

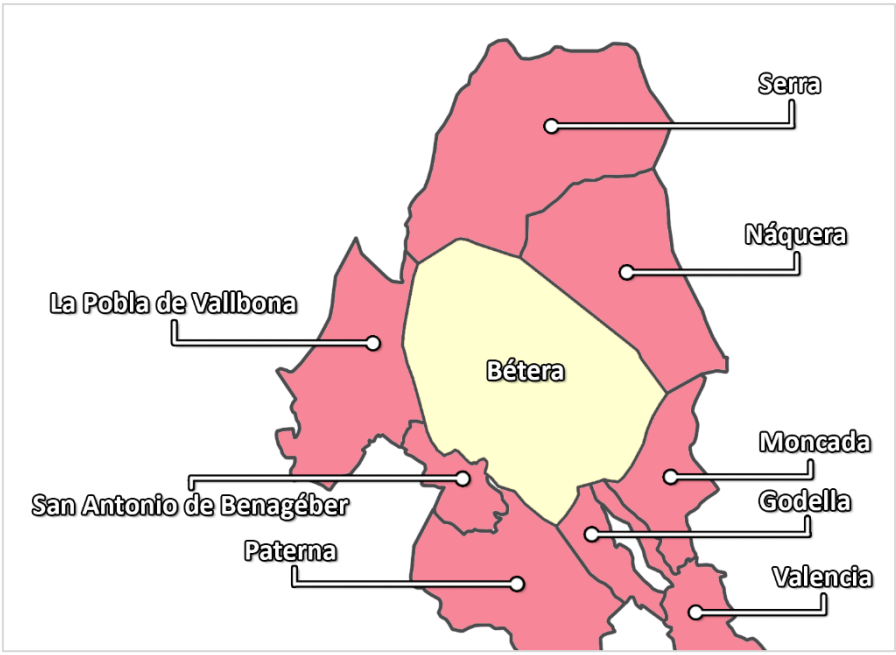


Figura 3.2. Límites Bétera.

4. Información del Castillo

El Castillo de Bétera es de origen medieval, se encuentra restaurado (Fig. 4.1), y con expediente incoado de declaración de Conjunto de Interés Histórico Artístico de carácter Local de junio de 1982.

Figura 4.1. Castillo de Bétera en la actualidad.

Tiene también la consideración de Bien de Interés Cultural Valenciano. Se encuentra Bajo la protección de la Declaración genérica del Decreto de 22 de abril de 1949, y la Ley 16/1985 sobre el Patrimonio Histórico Español.

El Castillo es el edificio que, junto al campanario de la Iglesia, más caracteriza a Bétera, dándole una apariencia física muy singular. Todas las entradas al pueblo están presididas por estos dos edificios.

Es de origen musulmán y, posiblemente, su origen fuera la torre más alta, la del reloj (Fig. 4.1). Edificado sobre una pequeña loma, que en aquel tiempo era la que más cerca estaba de la población, bastante reducida entonces, hoy en día se encuentra en el centro del casco antiguo de ésta.

Figura 4.2. Castillo de Bétera en la antigüedad (Fuente: Jorge Alonso Berdoza).

A pesar de que actualmente no se encuentra en la parte más alta del pueblo, desde cualquiera de sus torres puede observarse todo el término de Bétera, la Sierra Calderona y el mar. Por tanto, tiene una situación estratégica para cumplir la misión de cualquier fortaleza de la época en que se construyó.

Existen noticias de que el Castillo fue destruido el año 1364, y reedificado posteriormente, aunque parece que aquella destrucción sólo afectaría a sus muros y no a las torres existentes. Los materiales utilizados en la construcción son mampostería y argamasa para las zonas inferiores, y tapial para los muros y las torres.

Las piedras de canto se utilizaban generalmente para el interior. Excepcionalmente se utilizaba el ladrillo para reforzar algún ángulo. El acabado exterior estaba enlucido a capas, según la tradición árabe.

El castillo en todo su conjunto presenta un único recinto, de planta hexagonal de 1.480 m², ligeramente irregular, dado que los muros se sostienen sobre un terreno calcáreo que debió de mostrarse escabroso; también se debe esta irregularidad a la disposición de los muros, los cuales se encuentran en equilibrio con la ladera del terreno.

Con la finalidad de contrarrestar el desnivel, tanto los muros como las torres van asentadas sobre un zócalo en talud que se fue alzando hasta alcanzar la parte superior del escalón natural. Entre aquel y el muro hay una separación intermedia, que a la parte nordeste se encuentra eliminada por construcciones modernas.

En la zona occidental la plataforma avanza considerablemente, para dar lugar al inicio de dos baluartes que se sitúan a la entrada, y entre ellos queda un espacio hueco que, junto con el foso que se supone que existía, quedaba asegurada la defensa del acceso al interior. Las torres se encuentran distribuidas de forma indeterminada entre los lados y los ángulos, pero queriendo aproximarse hacia la zona norte, ya que, lógicamente, era la principal y necesitaba una mayor defensa.

El castillo se construyó con seis torres, dos de base semi-elíptica y cuatro de planta rectangular. De todas las torres destaca, sin duda por su gran volumen, una de planta rectangular y alzada tronco-piramidal, que corresponde a la torre señorial o del homenaje, con una altura de más de veinticinco metros desde la base, y ocho y seis metros respectivamente de cada lado.

En 1897 le instalaron un reloj con una campana, que daba las horas. En 1963 la torre de planta rectangular de la parte frontal del castillo se hundió junto con el tramo de muro que coincidía con la torre semi-ovalada del nordeste, de la que en esta fecha sólo quedaba una parte de los laterales. La torre del sudeste, de base rectangular, ya se había caído, pero no se sabe exactamente cuándo.

Todas las torres tienen almenas acabadas en pequeños prismas puntiagudos. La entrada al interior del castillo-palacio se encuentra en la zona occidental, y se hace pasando entre medio de los dos baluartes mencionados más arriba, encontrándose abrigada por la torre del homenaje y de una de las otras a la izquierda. Para poder entrar había una escalinata, la cual sustituía a la rampa original.

A continuación se encontraba otra puerta, y después de traspasar ésta con su vestíbulo, un arco ojival de austeras líneas góticas, del siglo XV, da paso a un espacioso patio central de forma cuadrangular, que debería ser el antiguo patio de armas. Este patio servía de acceso a

muchas dependencias, normalmente destinadas a servicios, en una época en la que el castillo era residencia temporal de los Barones de Bétera. A la izquierda, un arco de medio punto, hasta su restauración, era la entrada a la capilla. Enfrente, según se entra, nos encontramos una antigua galería o mirador con arcos de medio punto, que después se tapiaron para utilizar este espacio como aulas de formación. A la derecha podemos observar un vistoso arco ojival, análogo al anterior, dando paso a un vestíbulo que llega al salón Noble, después tapiado, y a la gran escalera que conducía a las plantas superiores. Esta zona del palacio constituía la residencia del señor feudal, el Barón de Bétera.

Hay una leyenda que asegura la existencia de un túnel bajo tierra, que va desde el castillo a la torre Bofilla, a más de tres kilómetros de la población. A pesar de que parece que esta leyenda es científicamente errónea, tiene un cierto fundamento, al haberse descubierto en algunas reformas de viviendas de la población tramos del supuesto túnel. En realidad, parece que de lo que se trataba era de antiguas salidas de emergencia del castillo, desde dónde se podía acceder al exterior.

Aunque no se conoce la fecha exacta de la construcción del castillo, por sus características, se sabe que es obra árabe (Fig. 4.3). También en el año 1238, cuando Bétera, junto con Paterna y Bofilla, se rindieron a Jaume I, la crónica habla de un castillo en nuestra población.

Figura 4.3. Castillo de Bétera en la antigüedad 2 (Fuente: Jorge Alonso Berdoza).

Desde su conquista perteneció a la Orden de Calatrava. En 1329 se vinculó a la Baronía de Bétera. Hacia finales del siglo XIV fue reconstruido, porque Pere IV lo mandó destruir el año 1364, a causa de su apoyo a las tropas unionistas.

En 1437 fue donado de manera perpetua a la familia Boïl. Posteriormente pasaría a la familia Rocafull y, finalmente, se vincularía a la familia del Marqués de Dos Aguas.

En el siglo XVI el castillo pierde su valor defensivo. Es decir, de ser una fortaleza pasa a tener un carácter palaciego. A partir de este momento empieza a sufrir cambios, para adaptarse a las necesidades de sus ocupantes. Las transformaciones externas están determinadas, sobre todo, por las construcciones realizadas delante de los muros, las cuales ocultaron manifestaciones más puras en el aspecto artístico, por ejemplo un ventanal gótico. Antiguamente recaía en la parte superior del muro nordeste, pero después sólo podía observarse desde el interior del castillo, pues se realizó una nueva construcción que aprovechaba el espacio intermedio del vestíbulo y lo ocultaba.

También desapareció el terraplén de la fortaleza. El interior del castillo ha sufrido tantas transformaciones que sólo se pueden apreciar pequeños elementos góticos, generalmente del siglo XV. Entre las alteraciones más frecuentes cabe destacar las nuevas paredes, la ampliación y reducción de puertas y ventanas, el relleno de los sótanos, los cuales debieron ser, antiguamente, la mazmorra y la bodega.

En la zona del muro exterior de la capilla sobresalen dos ménsulas de piedra que parecen indicar la antigua ubicación de un matacán, a pesar de que su situación, tan cercana a la torre Quinta parece desvirtuar su función. Al parecer sostenía una especie de jaula donde se encerraba, a la vista de todos y sobre la muralla, a quien merecía un castigo.

Durante las obras de rehabilitación se encontraron unas galerías excavadas durante la guerra civil, en la base maciza del castillo, construidas como refugio, los accesos de las cuales habían quedado ocultos. Estas galerías se introducían hacia el centro de la base del castillo, entrando por la plaza de la calle Andreu Fresquet y saliendo por debajo de la capilla, actual biblioteca.

Parece que iban buscando el supuesto túnel que comunicaba con la Torre Bofilla. Actualmente, y después de la rehabilitación, son utilizadas como salidas de emergencia del salón de plenos del Ayuntamiento, ubicado allí. Se cree que fueron los Barones de Bétera los que abrieron el actual acceso principal (entre las torres Primera y Sexta) y eliminaron los taludes existentes en la base del castillo en esa zona. De esta forma, se perdieron definitivamente los accesos de la rampa posterior.

El último señor del castillo, Vicente Dasí Lluesma (Fig. 4.4), hizo donación de la Casa-Castillo a la Junta de Monts i Senyoriu Territorial de Bétera el 16 de julio de 1888, con las siguientes condiciones:

1. Que fuera utilizado como escuela de párvulos, o como Hospital en caso de epidemia u otros análogos. La enseñanza o la asistencia sanitaria correría a cargo de religiosas.
2. La Junta de Monts correría con los gastos de mantenimiento del Asilo, así como con el sueldo de las monjas que trabajaran allí por esta causa.
3. Cuando dejara de utilizarse, quedaría sin efecto la donación y volvería al Señor Marqués.

Figura 4.4. Vicente Dasí Lluesma (Fuente: Jorge Alonso Berdoza).

Al convertirse el castillo en Asilo, en 1888, se añadió una construcción más entre las torres Tercera y Cuarta, y se eliminó el muro que existía antes. Ésta es la última ampliación que sufrió el castillo, la cual sobresalía por delante de las torres y le quitaba el aspecto de fortaleza. Esta construcción no se hizo con muros de carga, sino con soportes, vigas y paredes en los cerramientos, enlucidos de barro.

Presentaba una planta al nivel del patio, habilitada para aulas, y dejaba la parte inferior cubierta, como corral para los animales, la cual era a su vez una rampa de acceso al castillo. Los arcos de piedra que constituían las entradas más antiguas quedaban, al lado de la escalera central, como un oscuro pasaje entre la planta baja y el gallinero.

La Junta de Monts cumplió las condiciones impuestas por el Señor Marqués de Dos Aguas pero, como consecuencia del deterioro, el castillo se hizo inhabitable y peligroso, por lo que en el año 1970 se construyó un colegio nuevo en otra ubicación y la Casa-Castillo volvió al marquesado.

El 5 de noviembre de 1981 el Ayuntamiento de Bétera, en sesión plenaria, inicia el expediente para la declaración de la Casa-Castillo de Bétera como Monumento Histórico-Artístico de carácter provincial, y el 2 de junio de 1982, la Dirección General de Bellas Artes, Archivos y Bibliotecas acordó tener por incoado el expediente de declaración de monumento histórico-artístico a favor de la Casa-Castillo de Bétera. El 28 de julio de 1982 se publicó en el B.O.E. número 179.

El 16 de marzo de 1983 Don Pascual de Rojas y Cárdenas, Marqués de Dos Aguas y propietario del castillo, hizo donación del mismo a favor del Ayuntamiento de Bétera.

El Ayuntamiento de Bétera restauró y consolidó la Casa-Castillo. El Ministerio de Obras Públicas y Urbanismo encargó a Francisco Jurado Jiménez la redacción del Proyecto Técnico de las obras de Rehabilitación de la Casa-Castillo de Bétera. Las obras comenzaron en julio de 1984, inaugurándose su remodelación el 9 de octubre de 1989.

El resultado de esta rehabilitación ha hecho que las dependencias del castillo en la actualidad realicen las siguientes funciones:

- El vestíbulo sigue utilizándose como tal; a la izquierda se encuentra la conserjería, a la derecha la escalera que conduce a la torre principal, a la vez que puede comunicar con el resto de dependencias de la primera planta, donde hay una gran aula de conferencias, las aulas de E.P.A. (Enseñanza Permanente de Adultos), y otras aulas con diferentes utilidades. Al final de este vestíbulo hay una escalera que, subiendo, nos lleva al aula grande del primer piso, actualmente sala de conferencias.

- Si volvemos al vestíbulo, enfrente y subiendo dos escalones, hay otro vestíbulo acristalado que hace de separación del patio cuadrado; a la izquierda, donde estaba la capilla, hoy se encuentra la biblioteca municipal, donde se puede ver la base de una torre; hacia la derecha, un pasadizo nos lleva a otro vestíbulo, también acristalado, que conforma el otro lateral del patio, desde donde se puede ir a un aula grande, que podría ser el salón noble, y hoy es el despacho del Gabinete Psicopedagógico municipal. También se puede llegar a otra dependencia, subiendo una escalera de cinco o seis escalones, que se conserva intacta, y que era utilizada como presión; en el mismo vestíbulo encontramos una escalera que baja y nos lleva a la calle.

- Por la parte de atrás del castillo, mirando a la calle de Boïl, se puede encontrar una gran sala de construcción moderna, la sala de plenos del Ayuntamiento, a la cual se puede acceder por una rampa de suaves escalones o por una escalera muy empinada.

5. Recursos y datos de partida

Para comenzar con la realización de este proyecto se han utilizado una serie de recursos que numeraremos a continuación y que los mismos han servido como base para la creación final del modelo del castillo.

5.1. Planos del castillo

El Ayuntamiento de Bétera ha proporcionado los siguientes planos del castillo en formato CAD. Estos contienen la estructura interna del castillo y fachada, son los siguientes: (Fig. 5.1), (Fig. 5.2), (Fig. 5.3), (Fig. 5.4), (Fig. 5.5), (Fig. 5.6), (Fig. 5.7), (Fig. 5.8) y (Fig. 5.9).

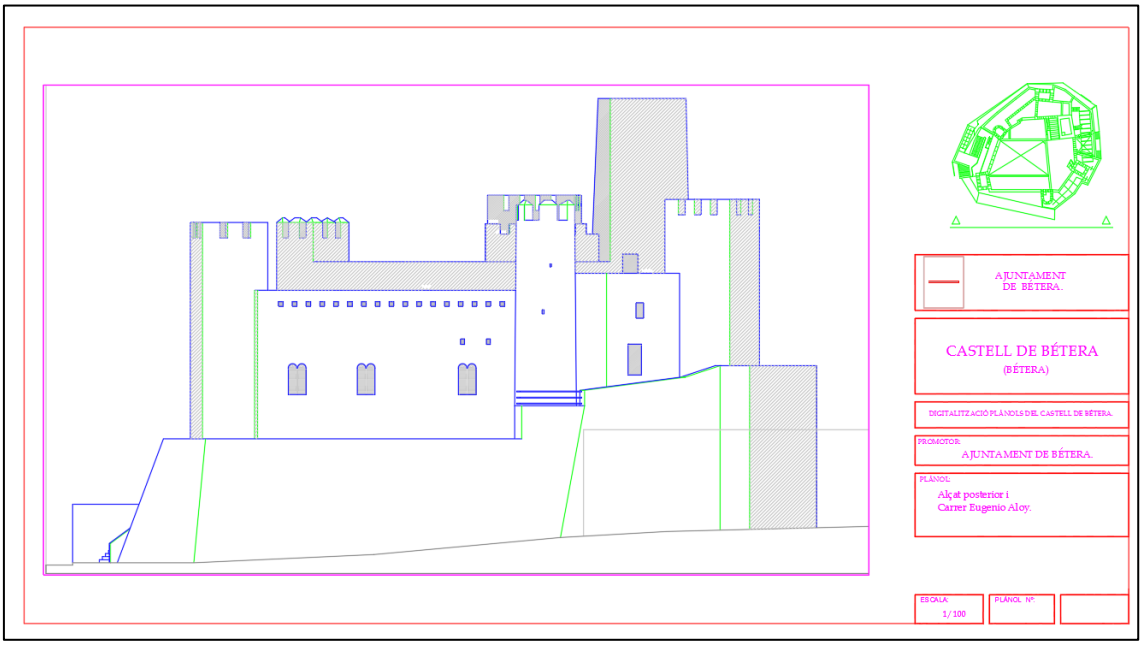


Figura 5.1. Alçat posterior, Carrer Eugenio Aloy (Fuente: Ayuntamiento de Bétera).

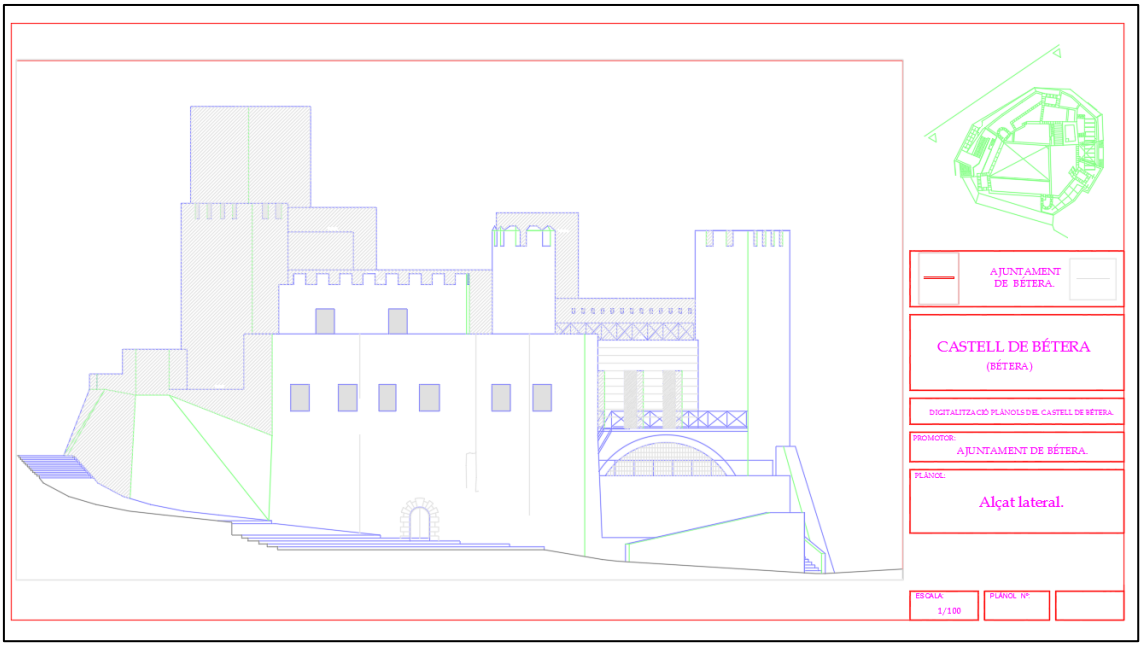


Figura 5.2. Alçat Lateral (Fuente: Ayuntamiento de Bétera).

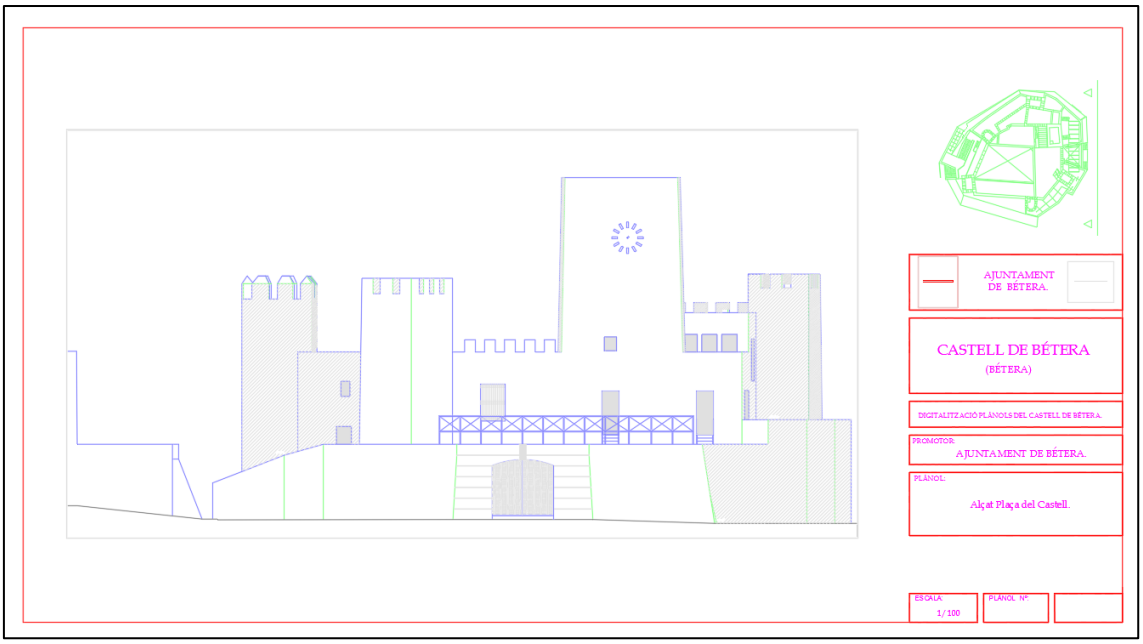


Figura 5.3. Alçat Plaça del Castell (Fuente: Ayuntamiento de Bétera).

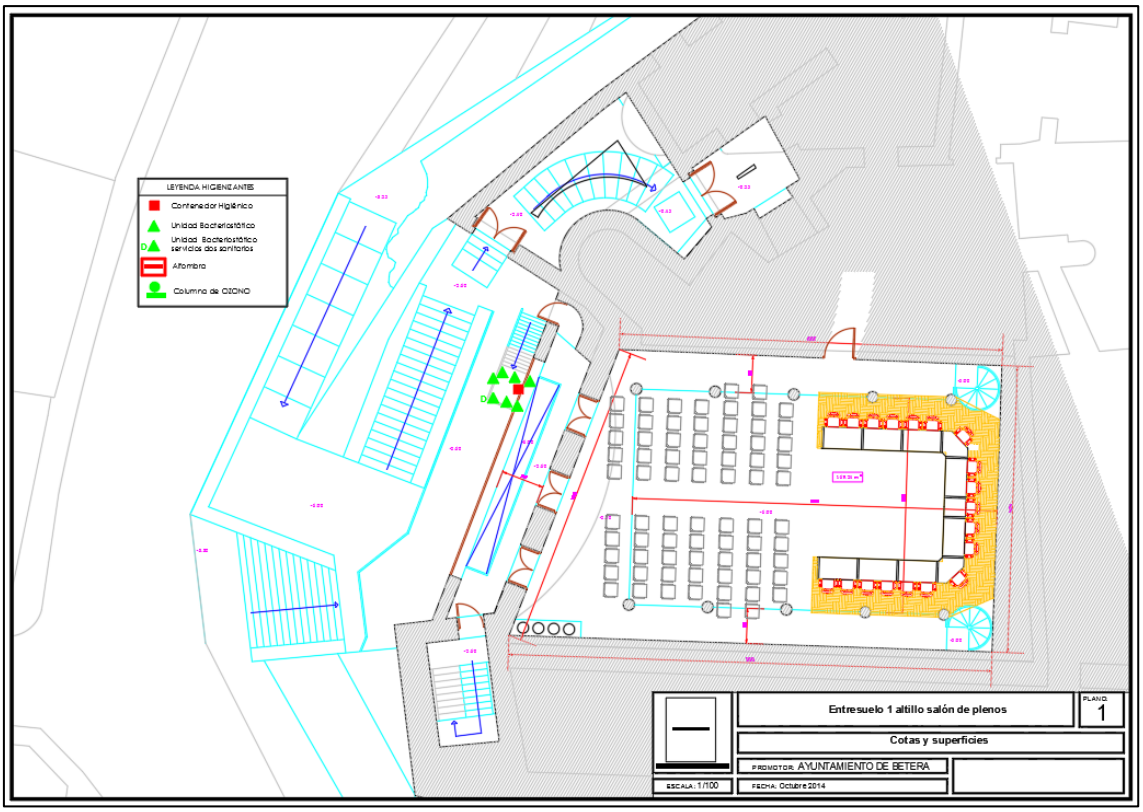
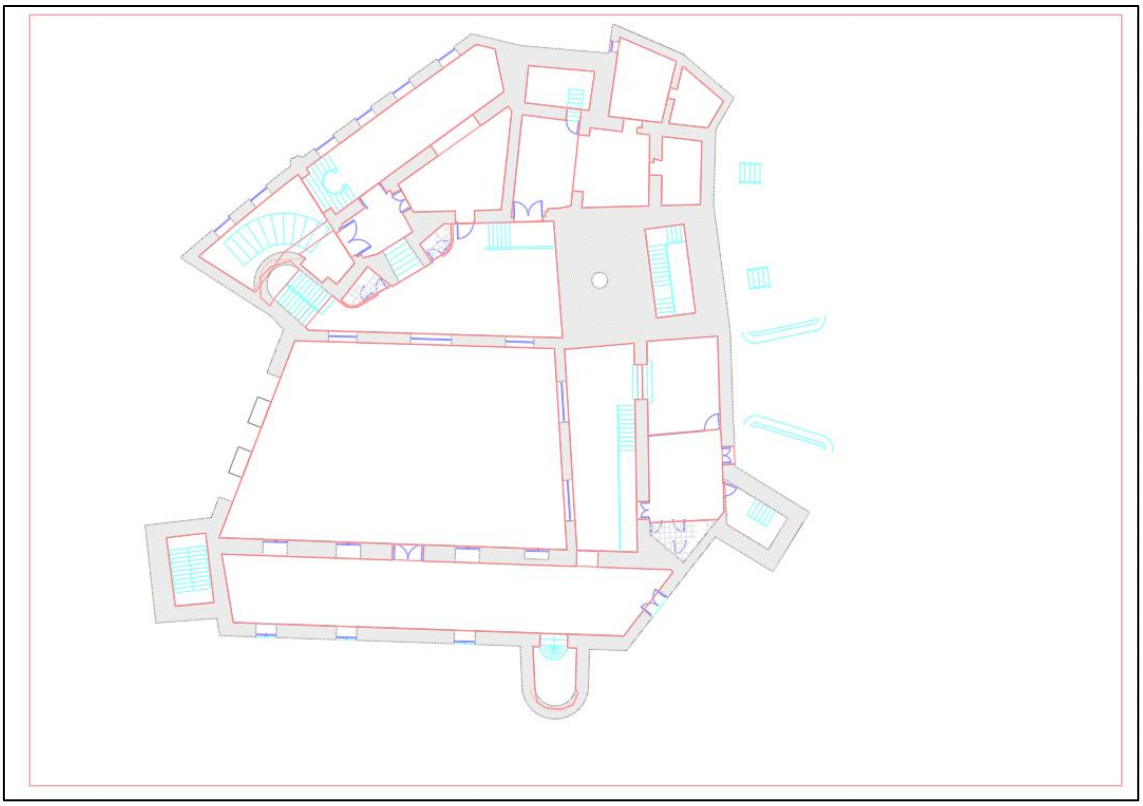


Figura 5.4. Entresolat 1 Altillo salón de plenos (Fuente: Ayuntamiento de Bétera).



1Figura 5.5. Entresolat 2 (Fuente: Ayuntamiento de Bétera).

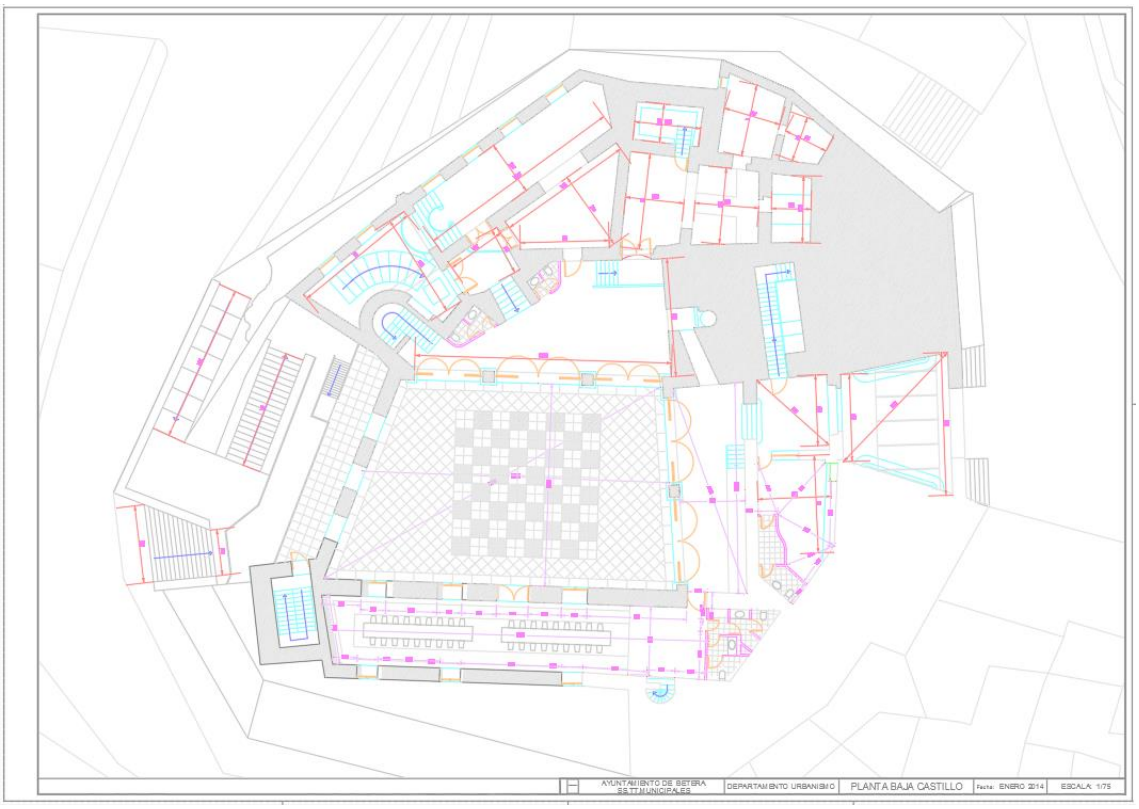


Figura 5.6. Planta Baixa Actualizada (Fuente: Ayuntamiento de Bétera).

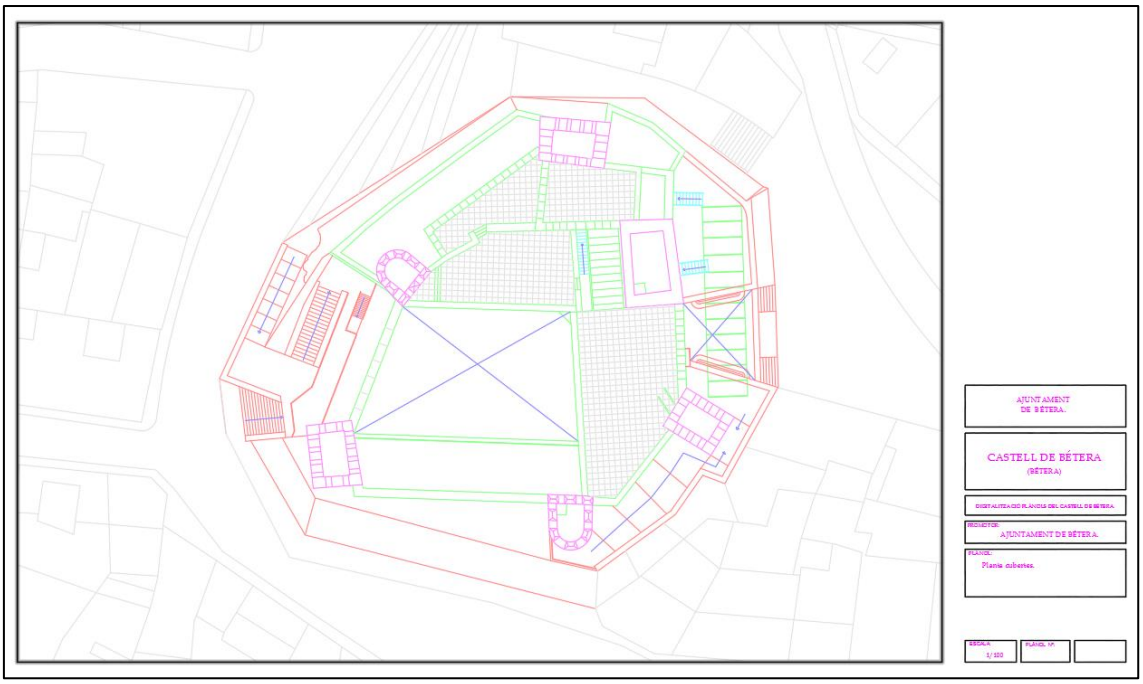


Figura 5.7. Planta Cubertes (Fuente: Ayuntamiento de Bétera).

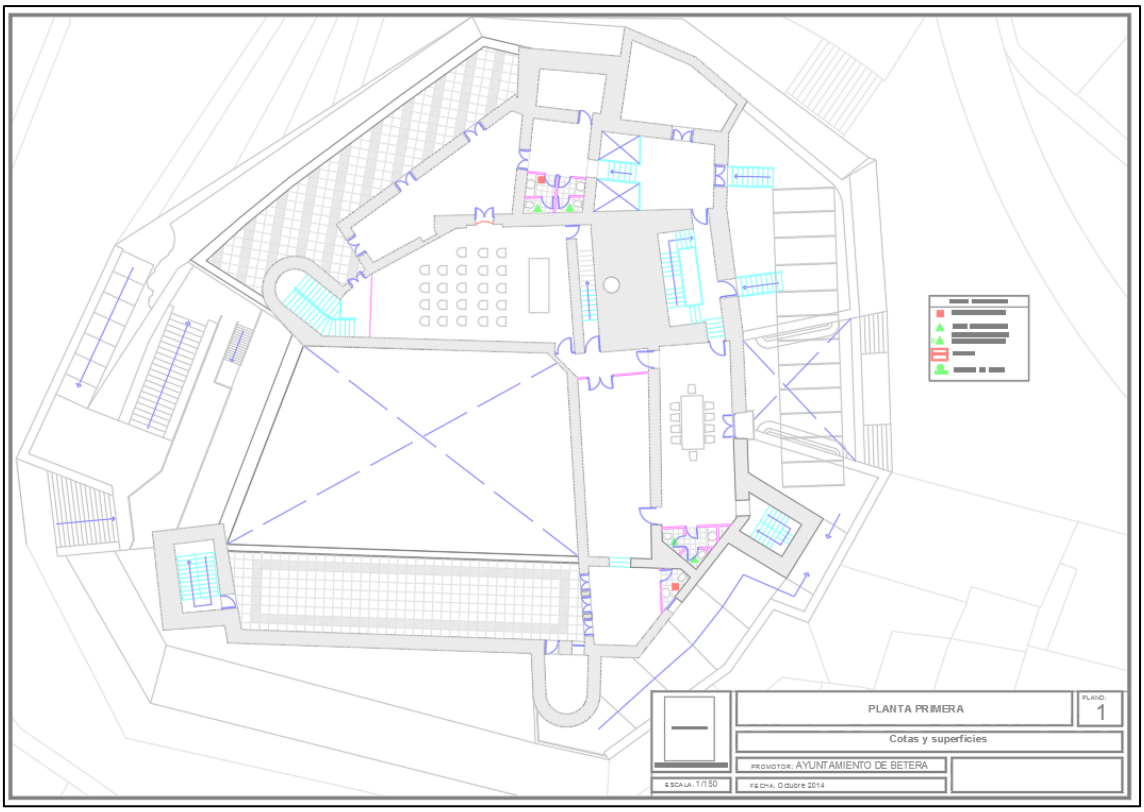


Figura 5.8. Planta Primera (Fuente: Ayuntamiento de Bétera).

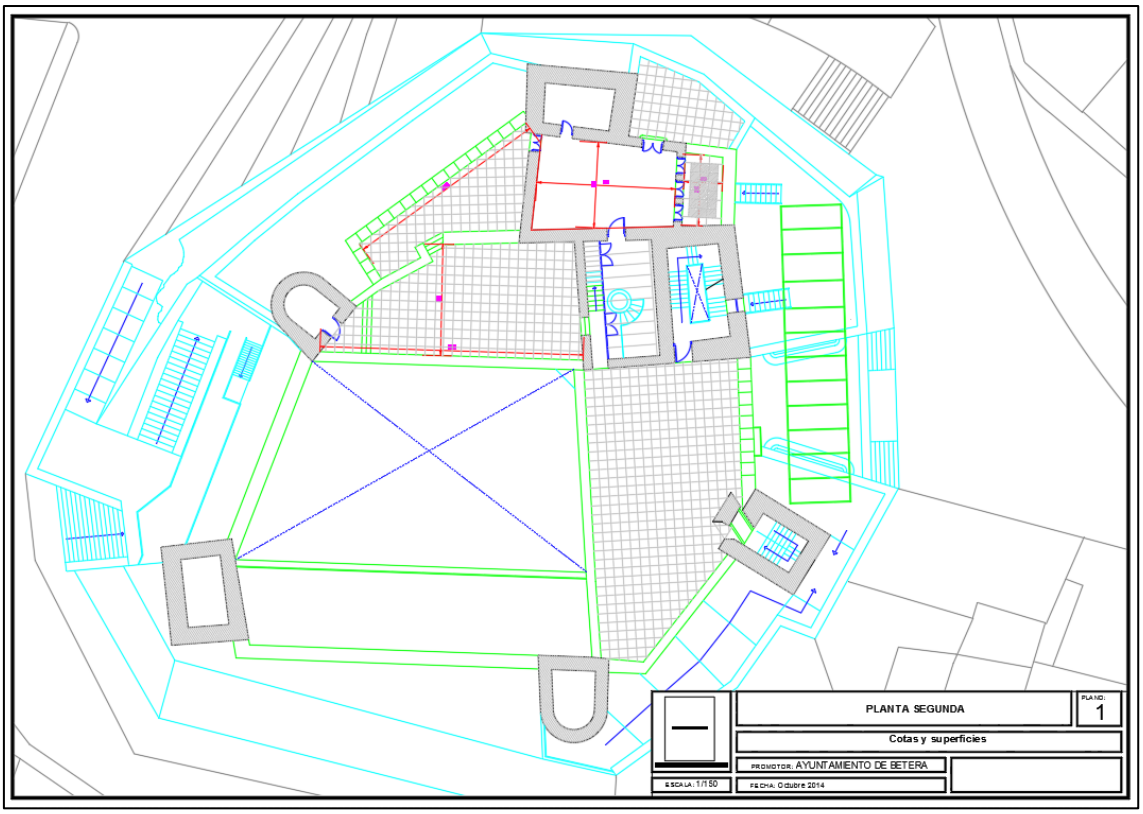


Figura 5.9. Planta Segona (Fuente: Ayuntamiento de Bétera).

5.2. Escaneado 3D

Se ha realizado un escaneado del patio interior del castillo que consta de 2 estaciones y 4 dianas situadas en diferentes planos y alturas del recinto (Fig. 5.10).

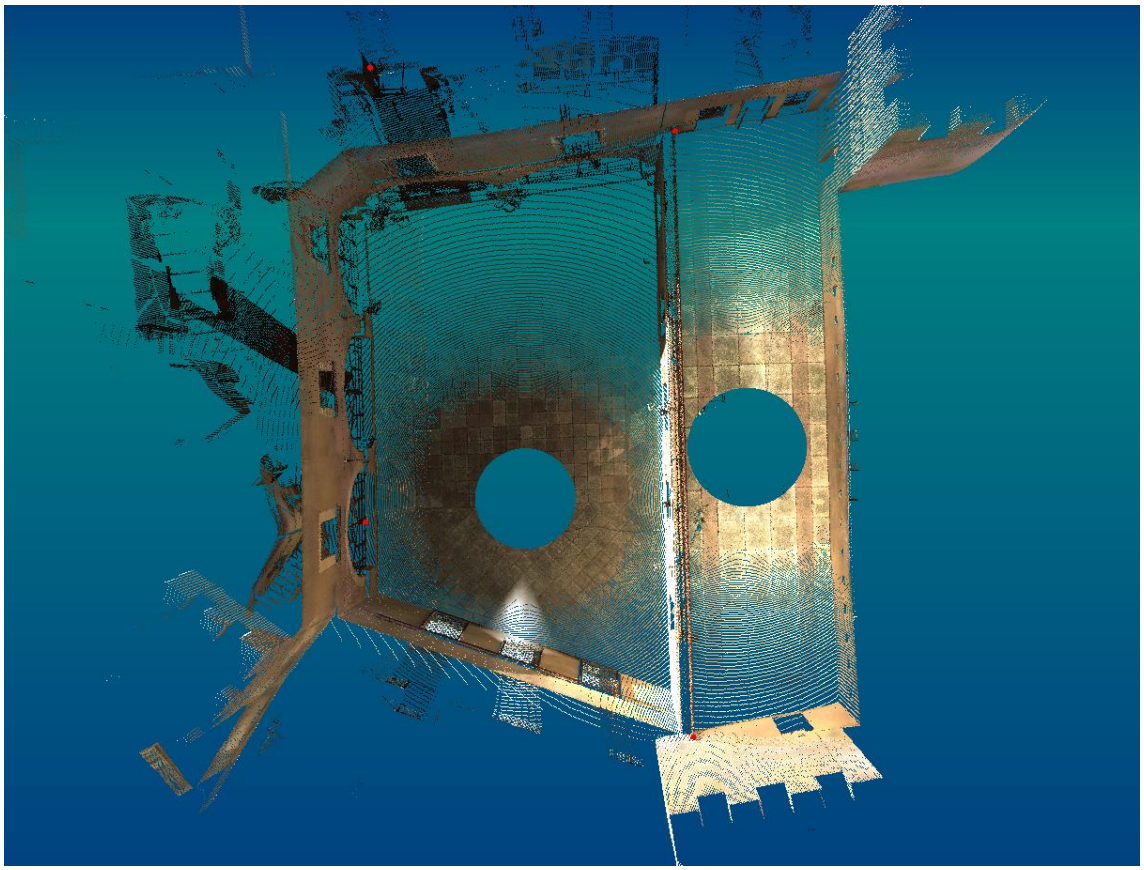


Figura 5.10. Planta nube de puntos registrada 3DVEM.

Para la primera estación se ha ajustado la resolución del escáner a unos 2 cm para una distancia de 10 m. Esta estación se ha situado en la parte central del patio.

La segunda estación se encuentra situada en una terraza del primer piso desde la cual se puede visualizar el patio interior y las dianas colocadas. La resolución utilizada para esta estación es la misma que para el caso anterior, 2 cm para una distancia de 10 m. Además se ha realizado un segundo barrido desde esta estación hacia una zona en la que se puede captar la iglesia de Bétera. Este segundo tiene una resolución de 2 cm para una distancia de 80 m.

Cabe mencionar que a pesar de obtener la información de la iglesia cercana al castillo, esta finalmente no se ha incluido en el presente proyecto, ya que se sitúa fuera del modelo y se ha optado por descartar esta información.

El aparato utilizado se corresponde al modelo ScanStation 2 de la marca Leica. En la Tabla 1 se pueden ver sus características técnicas según el manual extraido de la propia pagina web del fabricante.

Leica ScanStation 2 Performance Specifications		
Instrument type	Pulsed, dual-axis compensated, very-high speed laser scanner, with survey-grade accuracy, range, and field-of-view	
User interface	Notebook or Tablet PC	
Camera	Integrated high-resolution digital camera	
Accuracy of single measurement	Position*	6 mm
	Distance*	4 mm
	Angle (horizontal/vertical)	60 µrad/60 µrad (3.8 mgon/3.8 mgon) **
Laser spot size	From 0 – 50 m: 4 mm (FWHH-based); 6 mm (Gaussian-based)	
Modeled surface precision/noise	2 mm **	
Target acquisition	2 mm std. deviation	
Dual-axis compensator	Resolution 1", dynamic range +/- 5'	
Data integrity monitoring	Periodic self-check during operation and start-up	
Laser scanning system	Range	300 m @ 90%; 134 m @18% albedo
	Scan rate	Maximum instantaneous: up to 50,000 points/sec Average: dependent on specific scan density and field-of-view
	Scan density	< 1 mm max, through full range; fully selectable horizontal and vertical spacing; single point dwell capability
Laser class	3R (IEC-60825-1), visible green	
Lighting	Fully operational between bright sunlight and complete darkness	
Power supply	36 V; AC or DC; hot swappable	

Specifications subject to change without notice
See Leica ScanStation 2 Product Specifications for full technical data
* At 50 m range, one sigma
** One sigma

Tabla 1. Especificaciones del escáner láser Leica ScanStation 2.

5.3. Cartografía Vectorial Terrasit

Se ha realizado la descarga de la cartografía vectorial del municipio desde los servidores del portal http://terrasit.gva.es/ concretamente el archivo 46070-Betera-Vuelo2008.dxf (Fig. 5.11).

Figura 5.11. Cartografía vectorial Bétera 2008 (Fuente: http://terrasit.gva.es).

Con este archivo vectorial se han extraído las capas correspondientes a las viviendas, calles y curvas de nivel de la zona y posteriormente se ha efectuado un recorte de la zona del Castillo y sus alrededores.

Mediante los datos de las curvas de nivel se ha generado un modelo 3D del terreno. La metodología empleada ha sido la siguiente:

Importamos a 3ds Max únicamente las capas que contengan las curvas de nivel y activamos la opción Weld con un valor de 0,1 para que se suelden los vértices próximos evitando con esta opción aristas separadas (Fig. 5.12).

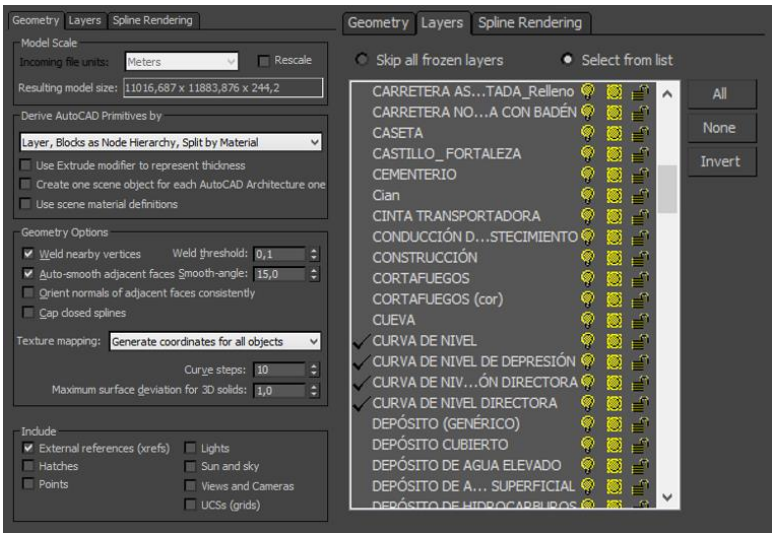


Figura 5.12. Importación capas curvas de nivel 3Ds Max.

A continuación se han unido las distintas capas cargadas para así crear un único objeto (Fig. 5.13).

Se ha realizado un recorte en la capa de las curvas para mantener únicamente las que quedan dentro de la zona del castillo (Fig. 5.14).

Seguidamente aplicamos a las curvas el comando Terrain, que se encuentra en el menú "Create", en el apartado de Compound Objects de 3D Max y ajustamos los parámetros de la herramienta en función del mayor o menor detalle que deseemos darle al terreno (Fig. 5.15).

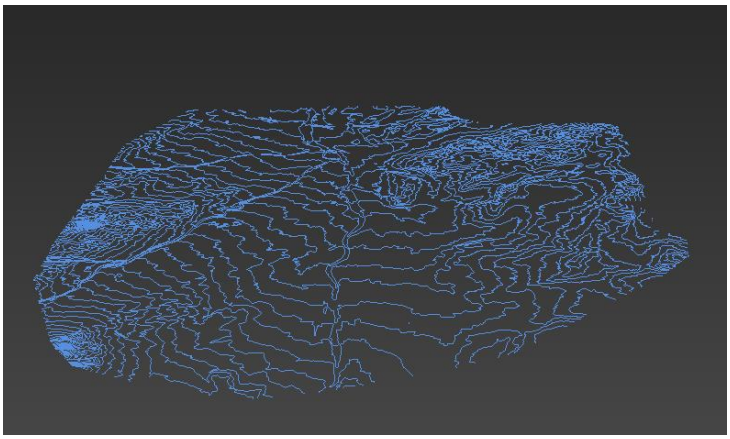


Figura 5.13. Curvas de nivel unidas en 3Ds Max.

Figura 5.14. Recortes curvas de nivel en 3Ds Max.

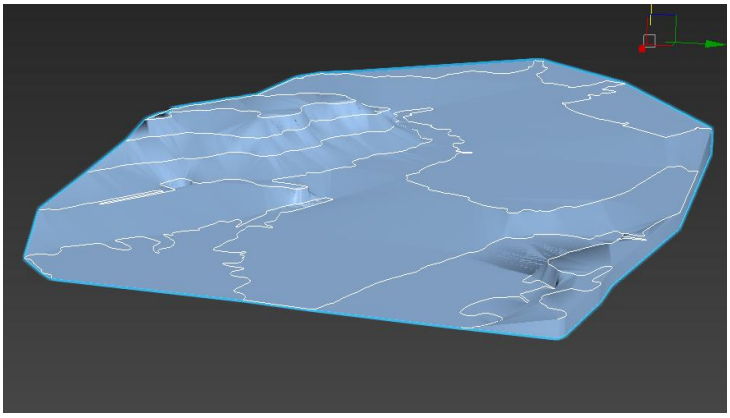


Figura 5.15. Terrain curvas de nivel en 3Ds Max.

Después de ajustar los parámetros observamos el terreno generado. Las caras que se crean se forman a partir de los puntos de las curvas de nivel. El algoritmo busca 3 puntos y crea una cara triangular hasta que se genera el terreno. Como se observa el número de caras generadas es muy elevado, por lo que no resulta de ninguna manera óptimo para el fin que deseamos (Fig. 5.16).

Para corregir éste problema podemos optar por dos opciones, la primera pasaría por crear un plano encima del terreno, y aplicar el comando Cloth al plano. Este se posará sobre el terreno como un trozo de tela y tomará la forma del terreno (Fig. 5.17).

La segunda opción pasa por ejecutar un plugin externo llamado Populate terrain, que se puede descargar gratis desde su página web http://populate3d.com/products/terrain/ pero solamente funciona con las versiones de 3ds Max desde el 2010 a 2014, por lo que tendremos que exportar el archivo a una versión anterior y ejecutar la herramienta desde el panel desplegable “Tools” situado en la parte superior del programa (Fig. 5.18).

Después aplicaremos un suavizado de malla y a continuación una optimización de malla reduciendo el número de polígonos, siendo éste el resultado (Fig. 5.19).

Finalmente utilizaremos esta última malla (Fig. 5.19) como base sobre la que situaremos los diferentes elementos que vayamos generando.

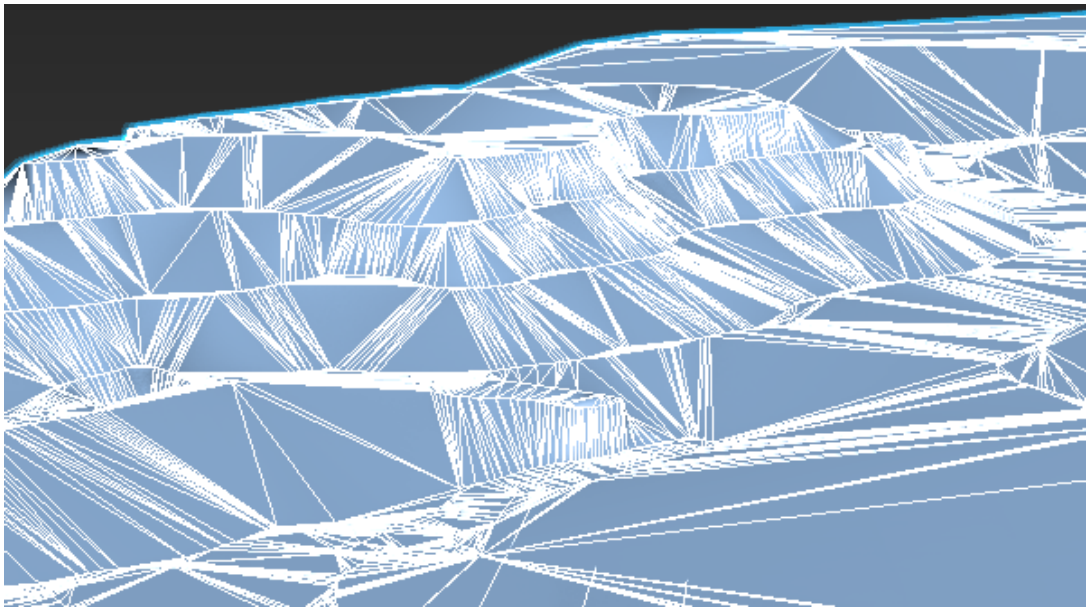


Figura 5.16. Triangulación curvas de nivel en 3Ds Max.

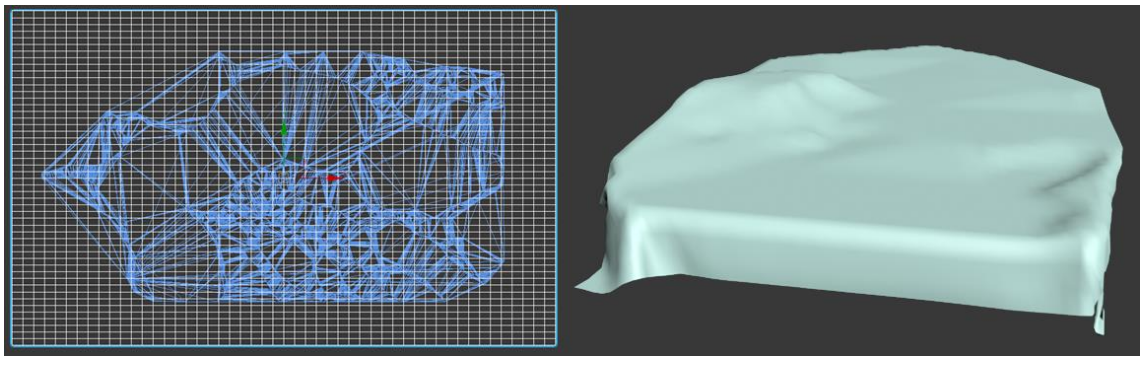


Figura 5.17. Cloth curvas de nivel en 3Ds Max.

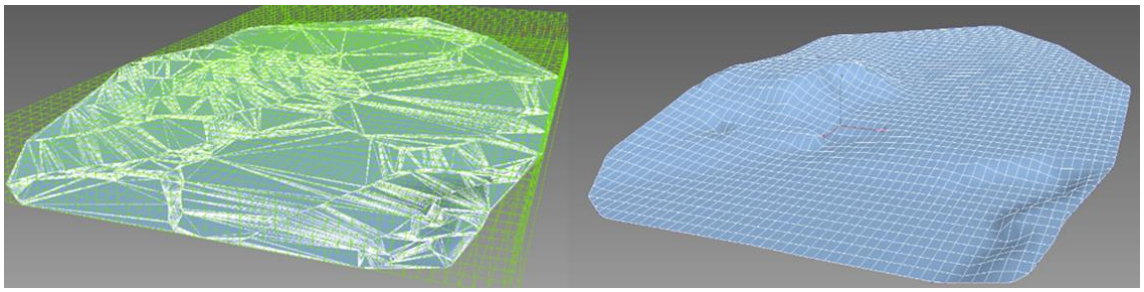


Figura 5.18. Populate Terrain.

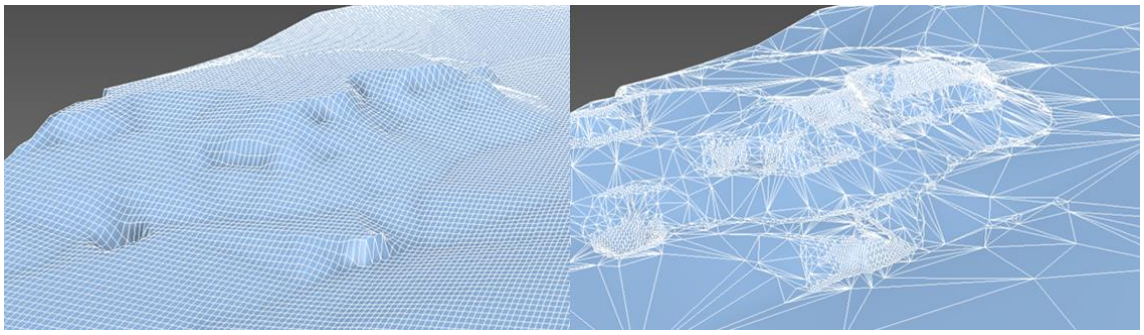


Figura 5.19. Terreno suavizado y optimizado.

5.4. Reportaje fotográfico

Para la creación de los edificios exteriores, así como para la posterior texturización de los objetos ha sido necesario realizar un amplio reportaje fotográfico de los exteriores del castillo.

Las imágenes se han realizado con el siguiente dispositivo móvil:

- Modelo: HTC One S
- Características de la cámara:
  - Objetivo: 28mm
  - Apertura focal: 2.0
  - ISO: Automático para todas la fotografías
  - Tamaño las imágenes: 8 megapíxeles
  - Exposición: Automático para todas las fotografías

Se ha optado por realizar las fotografías con el ISO y exposición en automático y su posterior retoque digital y corrección con el software Adobe Photoshop CS6.

El reportaje fotográfico se compone de 246 fotografías, en las que se ha capturado los edificios exteriores, la fachada del castillo, pavimento, aceras, las escaleras traseras, el patio interior, suelo del castillo y diferentes objetos entre otros, a continuación se muestra una pequeña selección de algunas de las imágenes tomadas (Fig. 5.20), (Fig. 5.21), (Fig. 5.22), (Fig. 5.23), (Fig. 5.24), (Fig. 5.25), (Fig. 5.26), (Fig. 5.27), (Fig. 5.28), (Fig. 5.29), (Fig. 5.30), (Fig. 5.31), (Fig. 5.32), (Fig. 5.33), (Fig. 5.34), (Fig. 5.35), (Fig. 5.36), (Fig. 5.37), (Fig. 5.38), (Fig. 5.39), (Fig. 5.40), además de poder apreciar la complejidad que ha supuesto modelado debido a la gran cantidad de elementos que forman el mismo.

Figura 5.20. Reportaje fotográfico 1.

Figura 5.21. Composición reportaje fotográfico 1.

Figura 5.22. Reportaje fotográfico 2.

Figura 5.23. Composición reportaje fotográfico 2.

Figura 5.24. Reportaje fotográfico 3.

Figura 5.25. Composición reportaje fotográfico 3.

Figura 5.26. Reportaje fotográfico 4.

Figura 5.27. Composición reportaje fotográfico 4.

Figura 5.28. Reportaje fotográfico 5.

Figura 5.29. Composición reportaje fotográfico 5.

Figura 5.30. Reportaje fotográfico 6.

Figura 5.31. Composición reportaje fotográfico 6.

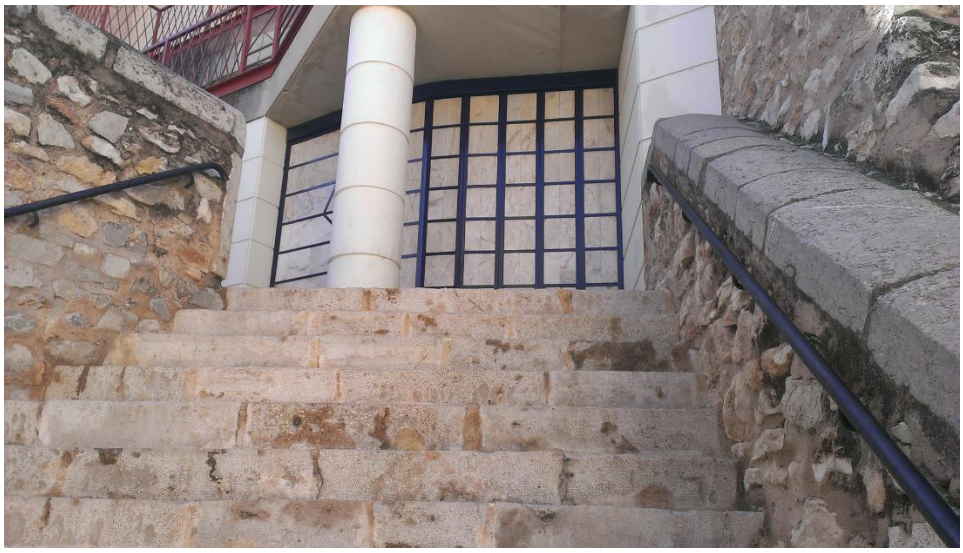


Figura 5.32. Reportaje fotográfico 7.

Figura 5.33. Composición reportaje fotográfico 7.

Figura 5.34. Reportaje fotográfico 8.

Figura 5.35. Composición reportaje fotográfico 8.

Figura 5.36. Reportaje fotográfico 9.

Figura 5.37. Composición reportaje fotográfico 9.

Figura 5.38. Reportaje fotográfico 10.

Figura 5.39. Composición reportaje fotográfico 10.

Figura 5.40. Reportaje fotográfico 11.

6. Modelado del castillo

El modelado es el proceso por el cual se crean los diferentes objetos tridimensionales que conforman la escena.

Para comenzar con el proceso de modelado se requiere de un esquema general del entorno para poder comenzar a trabajar. Además los recursos fotográficos sirven como referencias básicas para poder situar todos los objetos en su posición correcta.

En este caso los planos nos aportan la métrica de la planimetría y altimetría del castillo, además de proporcionar toda la estructura de salas del mismo.

Las imágenes junto con la cartografía descargada de Terrasit serán las encargadas de ayudarnos a modelar los exteriores del castillo.

6.1. Estructura interna del castillo

Para este apartado partimos de los planos proporcionados por el ayuntamiento de Bétera, de los cuales se han revisado las distintas capas, eliminado las que no eran necesarias y organizado correctamente.

Se ha comenzado situando en sus respectivas cotas las capas que corresponden al propio suelo de las distintas habitaciones, con el fin de empezar a estructurar internamente las distintas salas y poder hacernos una idea general del entorno interior.

En total el interior del castillo consta de plantas principales (Fig. 6.1):

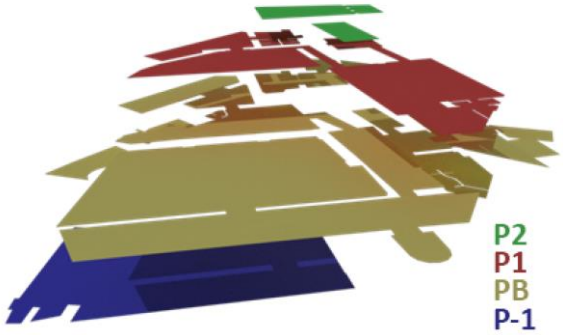


Figura 6.1. Plantas estructura interna.

Estas 4 plantas se han generado mediante extrusiones de los contornos de las paredes y finalmente se han cerrado formando los techos. También se han creado los distintos huecos de las ventanas, puertas y zonas conectadas entre las distintas salas (Fig. 6.2).

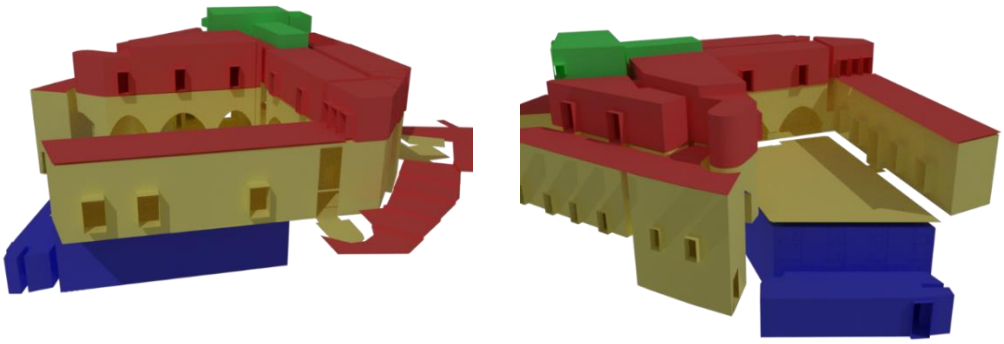


Figura 6.2. Extrusiones estructura interna.

También realizamos extrusiones en los interiores de los torreones hasta las respectivas cotas marcadas por los planos. Posteriormente ubicamos las escaleras interiores que permiten el acceso a las distintas plantas y a la parte superior de los torreones (Fig. 6.3).

Continuamos con el modelado de la fachada exterior, que consta de los muros exteriores y de cinco torreones (Fig. 6.4).

Por último añadimos la escalera de acceso posterior, la cual da acceso a la sala de plenos del ayuntamiento (Fig. 6.5).

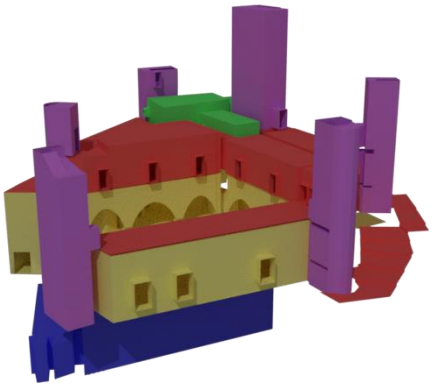


Figura 6.3. Escaleras y torreones.

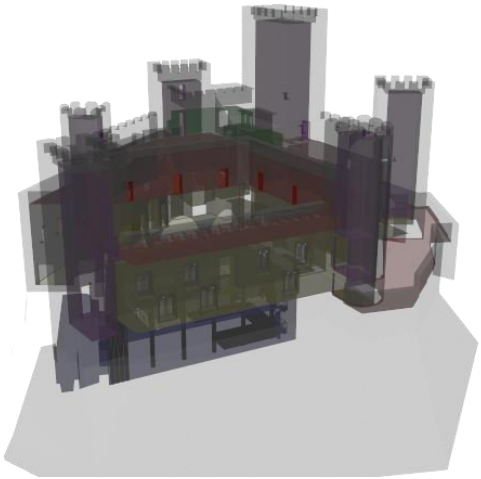


Figura 6.4. Fachada exterior.

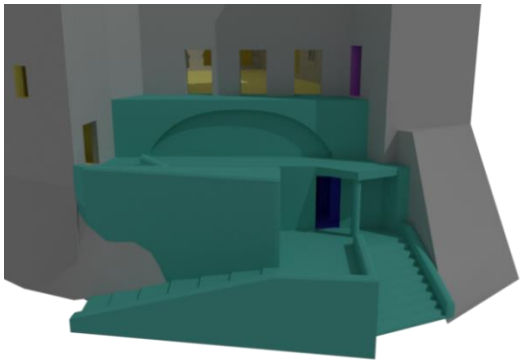


Figura 6.5. Acceso trasero.

6.2. Entorno Exterior

En este apartado nos topamos con la problemática de no disponer de datos altimétricos del entorno. Bien es verdad que disponemos de la cartografía vectorial de Terrasit y ésta nos ayuda a crear la forma del pavimento, aceras y diferentes viviendas que se encuentran en el exterior.

Por otro lado las curvas de nivel no solucionan la problemática de cometer errores en altimetría en edificios y el pavimento, cierto es que podríamos corregirlo tomando las pertinentes medidas nosotros mismos, dado que la finalidad de esta zona es crear un entorno cerrado para que el usuario no pueda salirse del modelo.

Por ello en este caso vamos a utilizar las curvas de nivel, que a pesar de darnos una mala precisión e incurrir en errores, al menos nos permitirán situar el pavimento y edificios de la forma más coherente posible, dejando a un lado la precisión.

Para el caso específico de los edificios además tenemos otra problemática añadida y es que no disponemos de las cotas de los tejados, reiterando en lo expuesto anteriormente, la finalidad de esta zona no es la precisión sino crear una envolvente lo más aparente posible; con datos precisos, sí que sería posible modelarlo con precisión. Por ello, hemos tenido que recurrir a una forma rápida de completar esta parte del modelado y tener unos edificios con proporciones aparentemente correctas para que no parezcan desproporcionadas (Fig. 6.6).

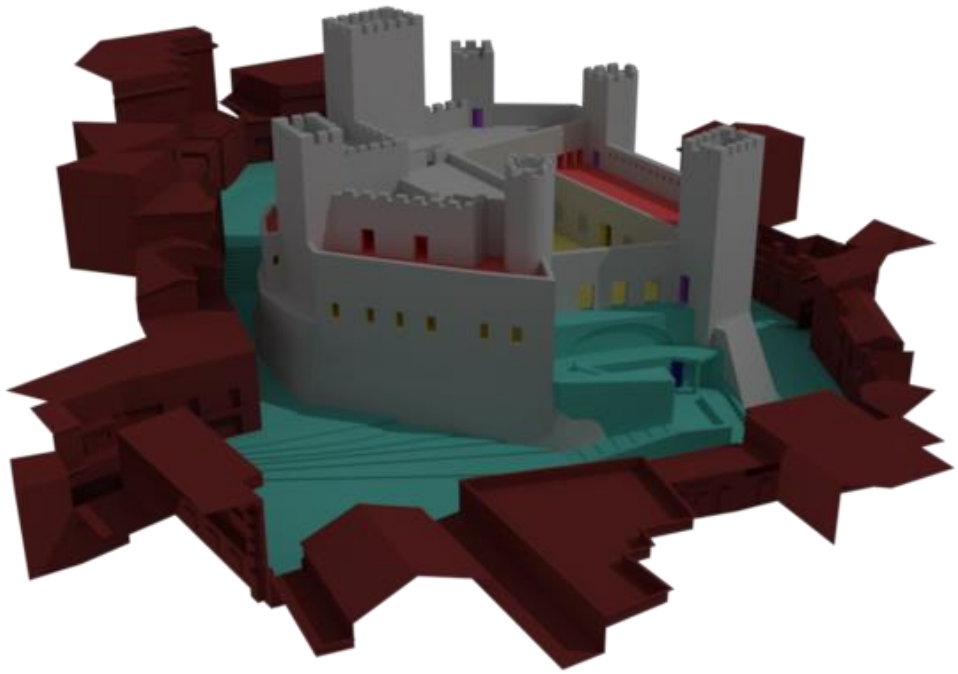


Figura 6.6. Entorno exterior.

La solución planteada consiste en optimizar el proceso de medición utilizando el reportaje fotográfico que utilizaremos para el texturizado posterior.

Simplemente en el momento que hemos realizado las fotografías hemos procurado capturar en la imagen toda la fachada, y si ésto no ha sido posible hemos unido las imágenes procurando no variar drásticamente el ángulo de la cámara.

Para la reconstrucción de las fachadas, se ha utilizado como base el polígono del edificio en cuestión. A continuación se ha colocado la imagen de la fachada en la base del polígono. Después se ha recortado y aplicado una rectificación plana a la imagen de la fachada (Fig. 6.7).

Como ejemplo vamos a tomar una fachada con un ángulo muy pronunciado que generará grandes errores (Fig. 6.7).

Las líneas rojas (Fig. 6.7) muestran claramente dónde debería estar el balcón y la puerta del garaje. Simplemente nos limitaremos a escalar la base de la fachada de la imagen respecto a la base del edificio que refleja la cartografía. Su altura no será correcta, pero nos permitirá tener unas distancias de referencia para modelar el total de los 27 edificios que componen el entorno.

Figura 6.7. Rectificación, escalado y errores.

Por último modelaremos y añadimos a su posición aproximada los elementos de mobiliario urbano, tales como los maceteros, farolas, contenedores, rejas, vallas, puertas, bancos, etc.

A continuación podemos ver los diferentes elementos situados y repartidos por el entorno (Fig. 6.8) y (Fig. 6.9).

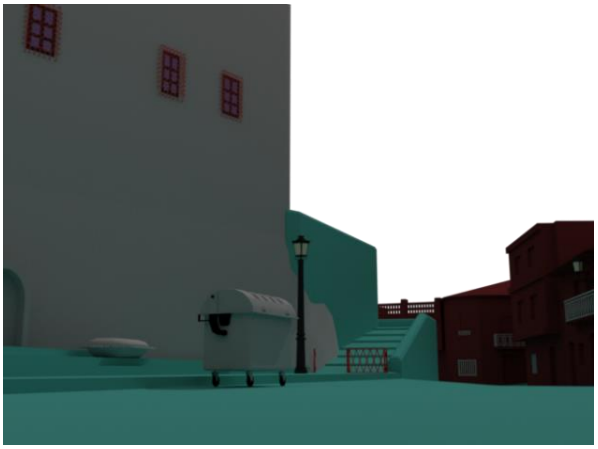


Figura 6.8. Objetos entorno 1.

Figura 6.9. Objetos entorno 2.

Figura 6.10. Objetos entorno 3.

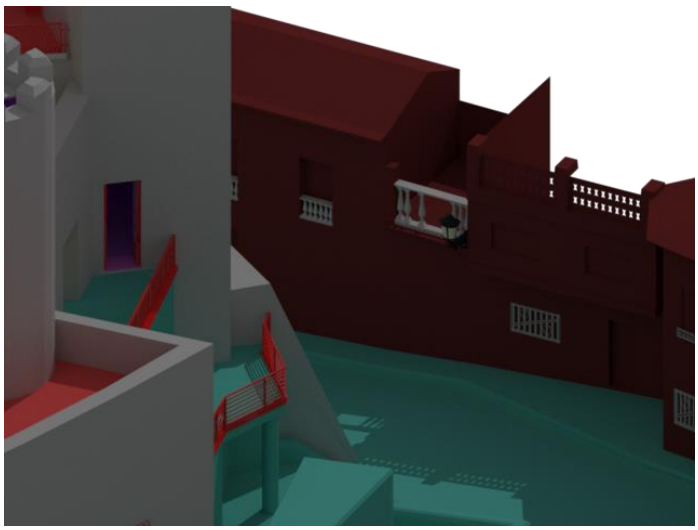


Figura 6.11. Objetos entorno 4.

6.3. Objetos y capas

En este apartado se muestra el listado de todos los objetos creados y la propia estructura organizativa de las capas que conforman el modelo. En total se compone de 147 objetos repartidos en 9 capas, desglosados de la siguiente forma: 46 en Entorno Exterior, 5 en Escaleras Traseras, 12 en Interior Torreones, 45 en objetos, 5 en P1, 12 en P-1, 4 en P2, 6 en PlantaBaja y 12 en Torreones.

- Entorno Exterior. (Fig. 6.12)
  - Ext_Asfalto
    - *Ext_Acera*
    - *Ext_AceraEntrada*
    - *Ext_Bordillos*
    - *Ext_Escaleras*
    - *Ext_Pavimento*
    - *Ext_Rampa*
  - Ext_Edificios
    - *Ext_Edificio1*
    - *Ext_Edificio2*
    - *Ext_Edificio3*
    - *Ext_Edificio4*
    - *Ext_Edificio5*
    - *Ext_Edificio6*
    - *Ext_Edificio7*
    - *Ext_Edificio8*
    - *Ext_Edificio9*
    - *Ext_Edificio10*
    - *Ext_Edificio11*
    - *Ext_Edificio12*
    - *Ext_Edificio13*
    - *Ext_Edificio14*
    - *Ext_Edificio15*
    - *Ext_Edificio16*
    - *Ext_Edificio17*
    - *Ext_Edificio18*
    - *Ext_Edificio19*
    - *Ext_Edificio20*
    - *Ext_Edificio21*
    - *Ext_Edificio22*
    - *Ext_Edificio23*
    - *Ext_Edificio24*
    - *Ext_Edificio25*
    - *Ext_Edificio26*
    - *Ext_Edificio27*
  - Ext_MobiliarioUrbano
    - *Ext_Bancos*
    - *Ext_BancosEntrada*
    - *Ext_Cadenas*
    - *Ext_Contenedores*
    - *Ext_Farolas*
    - *Ext_Macetero*
    - *Ext_MaceteroCircular*
    - *Ext_Pivotes*
    - *Ext_Puerta_Ext*
    - *Ext_Puerta*
    - *Ext_Uralita*
    - *Ext_VallaNegra*
    - *Ext_VallaNegra2*

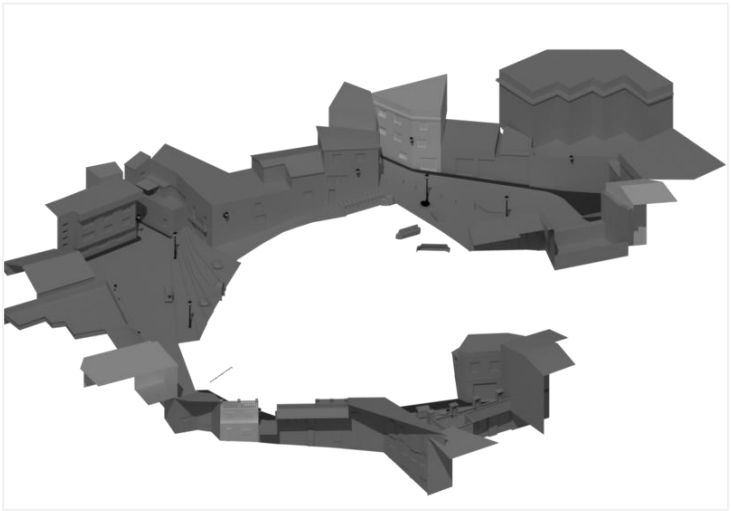


Figura 6.12. Entorno exterior.

- Escaleras Traseras. (Fig. 6.13)
  - *EScTra_Escaleras*
  - *EScTra_Muros*
  - *EScTra_Paredes*
  - *EScTra_Piedra*
  - *EScTra_Suelo*

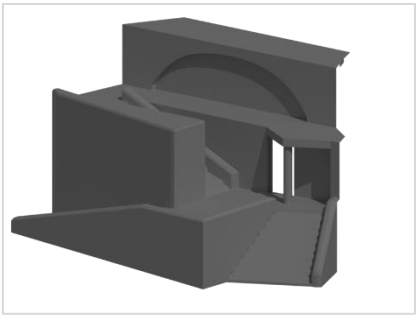


Figura 6.13. Escaleras Traseras.

- Interior Torreones. (Fig. 6.14)
  - *IntTorr_Esc_Torreon1*
  - *IntTorr_Esc_Torreon2*
  - *IntTorr_Esc_Torreon3*
  - *IntTorr_Esc_Torreon4*
  - *IntTorr_Esc_Torreon5*
  - *IntTorr_Esc_Torreon6*
  - *IntTorr_Torreon1*
  - *IntTorr_Torreon2*
  - *IntTorr_Torreon3*
  - *IntTorr_Torreon4*
  - *IntTorr_Torreon5*
  - *IntTorr_Torreon6*

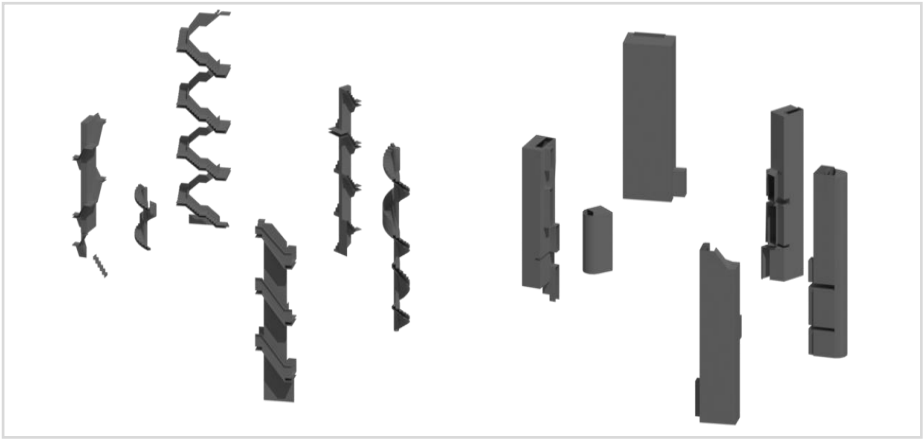


Figura 6.14. Interior torreones.

- Objetos. (Fig. 6.15)
  - Obj_Escaleras
    - *Obj_Escaleras_BarandillaEscalerasPB*
    - *Obj_Escaleras_PB*
    - *Obj_Escaleras_P1*
    - *Obj_Escaleras_AltilloBiblioteca*
  - Obj_Farolas
    - *Obj_Farolas_Cristal1*
    - *Obj_Farolas_Cristal2*
    - *Obj_Farolas_Cristal3*
    - *Obj_Farolas_Cristal4*
    - *Obj_Farolas_Cristal5*
    - *Obj_Farolas_Cristal6*
    - *Obj_Farola1*
    - *Obj_Farola2*
    - *Obj_Farola3*
    - *Obj_Farola4*
    - *Obj_Farola5*
    - *Obj_Farola6*
    - *Obj_Farolas_Bombilla1*
    - *Obj_Farolas_Bombilla2*
    - *Obj_Farolas_Bombilla3*
    - *Obj_Farolas_Bombilla4*
    - *Obj_Farolas_Bombilla5*
    - *Obj_Farolas_Bombilla6*
  - Obj_Puertas
    - *Obj_Puertas_CristalesMetalicas*
    - *Obj_Puertas_Madera*
    - *Obj_Puertas_Metalicas*
  - Obj_Tuberias
    - *Obj_TuberiaPatioInt*
  - Obj_Vallas
    - *Obj_Vallas_RejaVentana*
    - *Obj_Vallas_Barandillas*
    - *Obj_Vallas_EscaleraTerraza*
    - *Obj_Vallas_EstructuraTerraza*
    - *Obj_Vallas_PuertasTraseras*
    - *Obj_Vallas_RejaBalconPatio*
    - *Obj_Vallas_RejaVentanas*
  - Obj_Ventanas
    - *Obj_Ventanas_CristalesPatioInt*
    - *Obj_Ventanas_PuertaPatioInterior*
    - *Obj_Ventanas_VentanasYMarco*

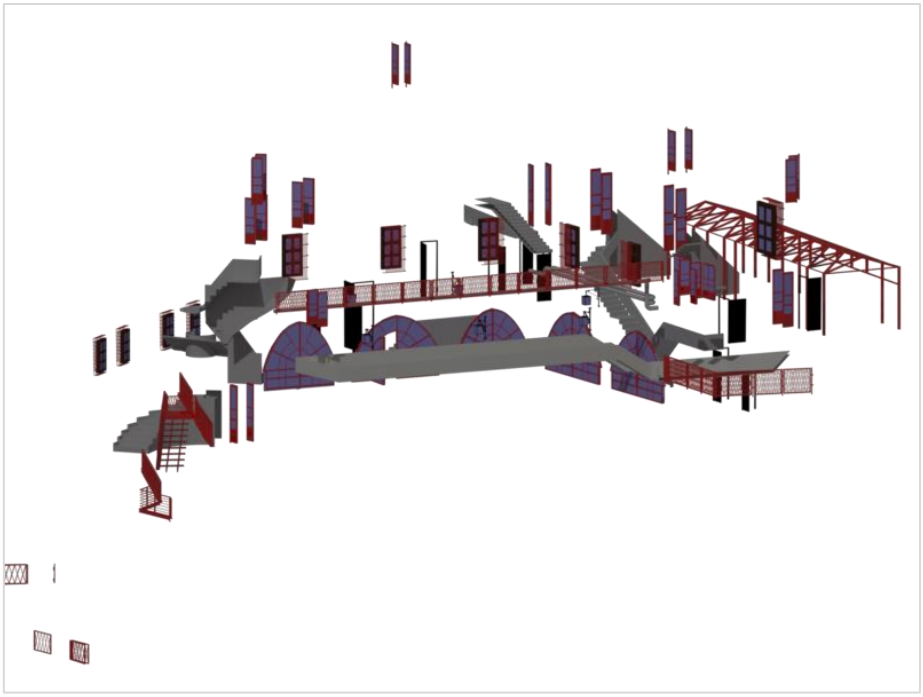


Figura 6.15. Objetos.

- P1 (Fig.6.16).
  - *P1_Muros*
  - *P1_Suelo*
  - *P1_SueloTerrazas*
  - *P1_Techos*
  - *P1_Zocalo*

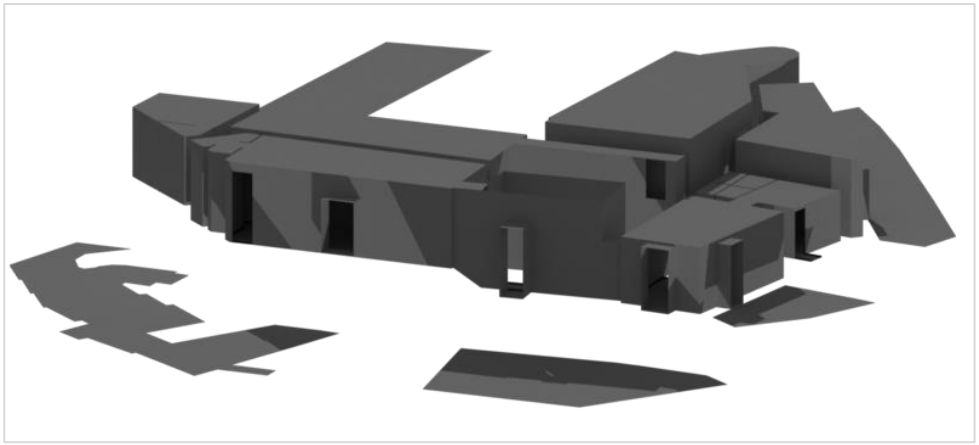


Figura 6.16. P1.

- P-1. (Fig. 6.17)
  - *P-1_Altillo*
  - *P-1_Entrada*
  - *P-1_EscaleraDch*
  - *P-1_EscaleraIzq*
  - *P-1_Habitacion*
  - P-1_Objetos
    - *P-1_Barandilla*
    - *P-1_FiguraCastillo*
    - *P-1_Mesa*
    - *P-1_Pilares*
    - *P-1_Tarima*
  - *P-1_Suelo*
  - *P-1_Techo*

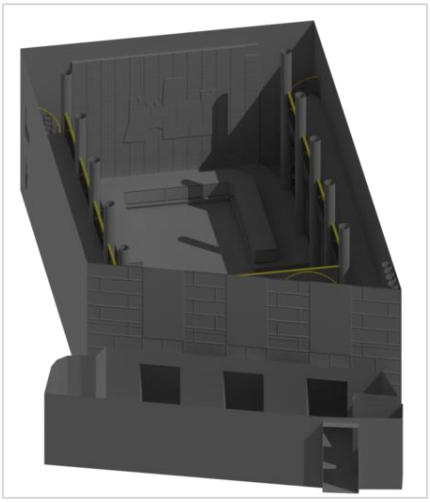


Figura 6.17. P-1.

- P2. (Fig.6.18)
  - *P2_Muros*
  - *P2_Suelo*
  - *P2_Techos*
  - *P2_Zocalo*

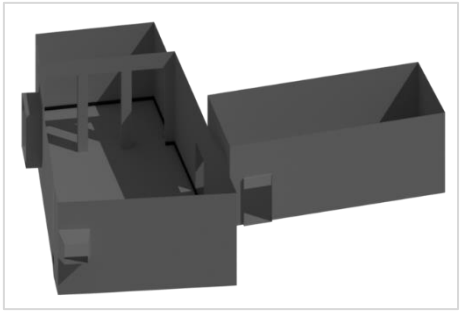


Figura 6.18. P2.

- PlantaBaja. (Fig. 6.19)
  - *PlantaBaja_Muros*
  - *PlantaBaja_Suelo*
  - *PlantaBaja_SueloPatioInterior*
  - *PlantaBaja_TechoEntresuelo*
  - *PlantaBaja_Techos*
  - *PlantaBaja_Zocalo*

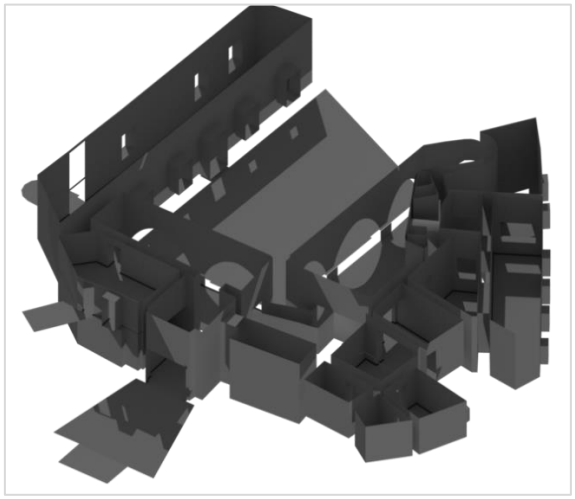


Figura 6.19. PlantaBaja.

- Torreones. (Fig.6.20)
  - *Torr_Muro_Entrada*
  - *Torr_Muro_Este*
  - *Torr_Muro_Inferior*
  - *Torr_PatioInterior*
  - *Torr_PatioInterior2*
  - *Torr_SuelosTerrazas*
  - *Torr_TorreonesOeste2*
  - *Torr_TorreonEste*
  - *Torr_TorreonEste2*
  - *Torr_TorreonOeste*
  - *Torr_TorreonReloj*
  - *Torr_VentanasMadera*

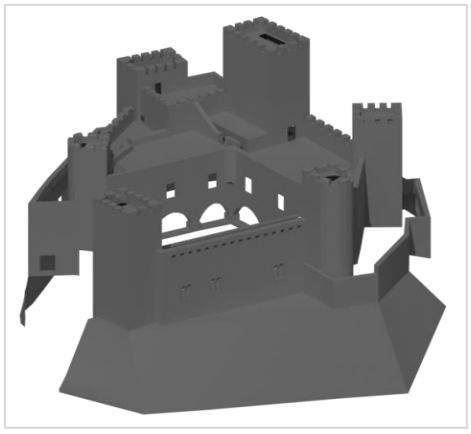


Figura 6.20. Torreones.

6.4. Modelo escaneado

Mediante el escaneado realizado del patio interior del castillo, gracias a éste vamos a reconstruir su superficie, creando un objeto sólido de menor resolución en formato .obj que es el óptimo para la importación a los distintos programas que utilizaremos posteriormente.

6.4.1. Registro

En primer lugar debemos realizar el registro de las nubes de puntos de los diferentes escaneados. Con ello conseguimos que éstas adopten el mismo sistema de coordenadas locales.

A continuación se han Exportado los distintos escaneados con el programa Leica Cyclone a formato pts, para posteriormente utilizar el programa 3DVEM – Register GEO para realizar el registro de manera automática.

Simplemente debemos ir a la pestaña “Datos” del programa e “Importar Ficheros”. Aquí seleccionaremos los dos archivos .pts de los escaneados que hemos realizado anteriormente y pulsamos el botón de Registrar.

Los parámetros aplicados para el registro han sido: (Fig. 6.21)

- Sin Estimación Robusta.
- Corrección lineal mínima 0.001m.
- Corrección angular mínima 3’’.
- Iteraciones máximas 7.
- Estaciones sin nivelar.
- Compensador de doble eje 1’’.
- Emparejado automático con una tolerancia de 0.010 m.

Como resultado del ajuste observamos que se ha realizado en 3 iteraciones y el error mínimo cuadrado es de 0.0006 m.

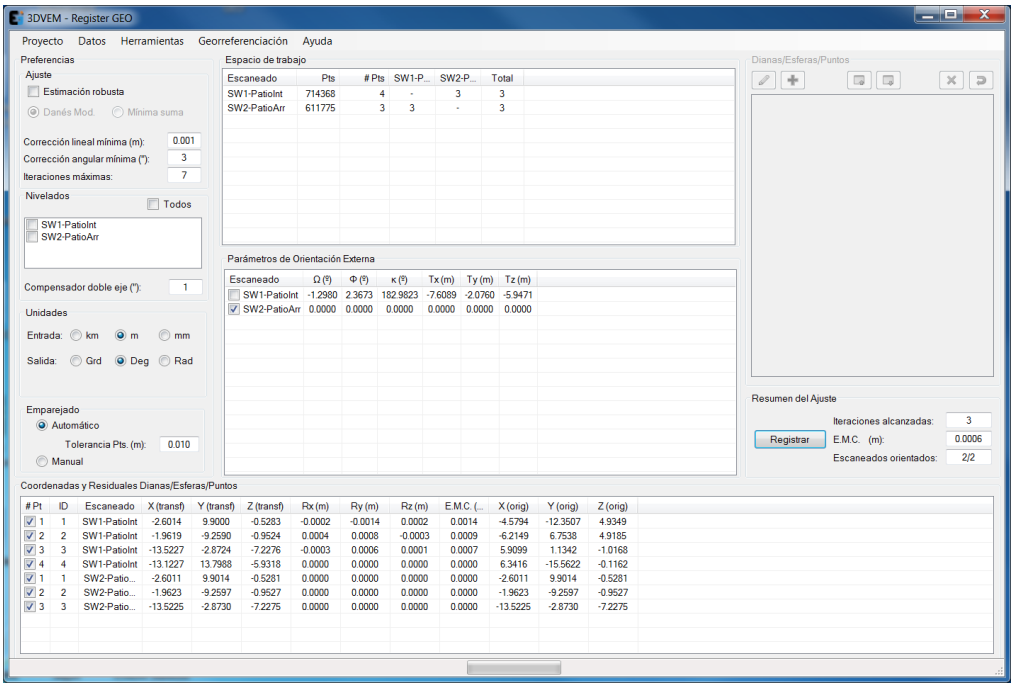


Figura 6.21. Registro.

Después seleccionamos la pestaña “Proyecto” y “Generar Informe”, se creará un archivo de texto donde nosotros seleccionemos con los resultados del registro. En la Tabla 2 podemos ver las coordenadas de las dianas.

SW1-PatioInt

Id	#Pt	X	Y	Z
1	1	-4.5794	-12.3507	4.9349
2	2	-6.2149	6.7538	4.9185
3	3	5.9099	1.1342	-1.0168
4	4	6.3416	-15.5622	-0.1162

SW2-PatioArr

Id	#Pt	X	Y	Z
1	1	-2.6011	9.9014	-0.5281
2	2	-1.9623	-9.2597	-0.9527
3	3	-13.5225	-2.8730	-7.2275

Tabla 2. Coordenadas de las dianas.

En la Tabla 3 se muestran los parametros de la orientacion externa, que corresponden a las rotaciones y traslaciones aplicados en este caso para el primer escaneado, respecto al segundo como fijo.

Escaneado	Ω (º)	Φ (º)	κ (º)	Tx (m)	Ty (m)	Tz (m)
☐ SW1-PatioInt	-1.2980	2.3673	182.9823	-7.6089	-2.0760	-5.9471
☑ SW2-PatioArr	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Tabla 3. Parámetros de orientación externa.

Y en la Tabla 4 encontramos los residuales de las dianas.

# Pt	ID	Escaneado	X (transf)	Y (transf)	Z (transf)	Rx (m)	Ry (m)	Rz (m)	E.M.C. (m)	X (orig)	Y (orig)	Z (orig)
☑ 1	1	SW1-PatioInt	-2.6014	9.9000	-0.5283	-0.0002	-0.0014	0.0002	0.0014	-4.5794	-12.3507	4.9349
☑ 2	2	SW1-PatioInt	-1.9619	-9.2590	-0.9524	0.0004	0.0008	-0.0003	0.0009	-6.2149	6.7538	4.9185
☑ 3	3	SW1-PatioInt	-13.5227	-2.8724	-7.2276	-0.0003	0.0006	0.0001	0.0007	5.9099	1.1342	-1.0168
☑ 4	4	SW1-PatioInt	-13.1227	13.7988	-5.9318	0.0000	0.0000	0.0000	0.0000	6.3416	-15.5622	-0.1162
☑ 1	1	SW2-PatioArr	-2.6011	9.9014	-0.5281	0.0000	0.0000	0.0000	0.0000	-2.6011	9.9014	-0.5281
☑ 2	2	SW2-PatioArr	-1.9623	-9.2597	-0.9527	0.0000	0.0000	0.0000	0.0000	-1.9623	-9.2597	-0.9527
☑ 3	3	SW2-PatioArr	-13.5225	-2.8730	-7.2275	0.0000	0.0000	0.0000	0.0000	-13.5225	-2.8730	-7.2275

Tabla 4. Residuales dianas.

Aquí podemos ver el resultado de las dos estaciones registradas (Fig. 6.22) con el programa 3DVEM - Viewer, Editor & Meter.

Figura 6.22. Registro estaciones y nube de puntos.

6.4.2. Reconstrucción de la malla

A continuación utilizaremos el programa MeshLab para cargar los archivos de las estaciones registradas y realizaremos un filtrado/borrado de los puntos que no sean válidos para la reconstrucción de la malla, tales como artefactos, objetos, puntos exteriores y puntos situados detrás de las ventanas, procurando dejar únicamente los puntos que conforman las paredes y estructura del patio central.

Como podemos ver (Fig. 6.23), se han eliminado todos esos puntos exteriores dejando como resultado una nube de puntos lo más limpia posible para posteriormente realizar una reconstrucción de la superficie.

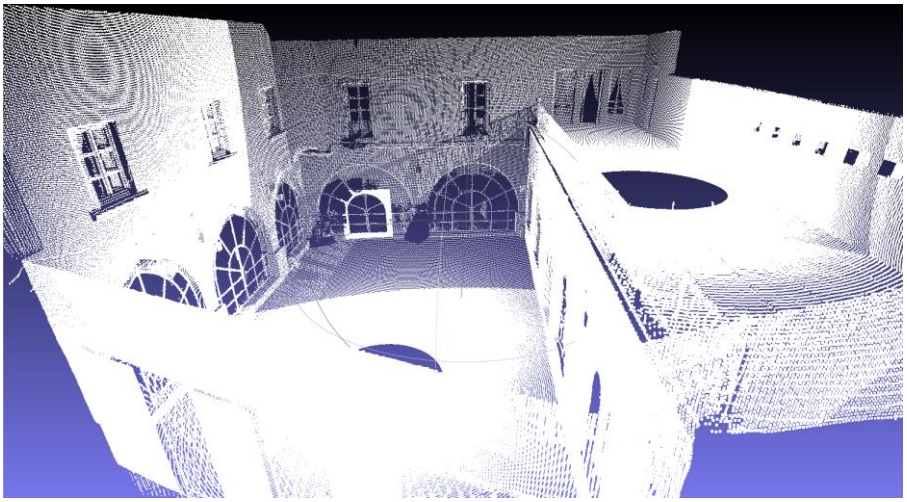


Figura 6.23. Nube de puntos filtrada en MeshLab.

Para comenzar con la reconstrucción de la superficie de la nube de puntos en primer lugar debemos realizar el cálculo de las normales de los puntos. Para ello en MeshLab debemos dirigirnos al apartado “Filters” -> “Point Set” y seleccionar el algoritmo “Compute Normals for point sets”

Este algoritmo calcula la orientación de la normal perteneciente a un vértice utilizando los demás puntos cercanos o vecinos de la nube de puntos. Realizar este paso es muy importante para que la orientación de las caras de la malla sea coherente y estén orientadas en la misma dirección, por lo que si no realizásemos este paso, podríamos tener problemas en la reconstrucción y encontrarnos con las caras de los polígonos desorientadas de forma aleatoria y se debería corregir de forma manual.

Para este algoritmo se ha utilizado un valor de 8 para el número de vecinos que influyen en el cálculo y un valor de 0 para el suavizado, siendo este valor el que propaga las normales calculadas a los puntos vecinos (Fig. 6.24).

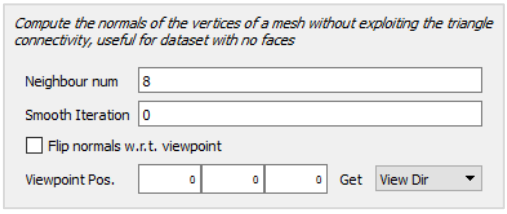


Figura 6.24. Cálculo de las normales para las nubes de puntos.

Dependiendo de la resolución o del detalle que queramos mantener podemos aplicar un muestreo de los puntos con tal de aplicar un filtrado a los mismos. Ésto nos permitirá reducir el número de polígonos de la superficie, así como el tiempo de cálculo de la superficie sacrificando detalles de la nube de puntos. En este caso concreto como queremos mantener la máxima resolución de la malla y se ha optado por no aplicar ningún filtro a la nube. Pero bien es verdad que si disponemos de un mayor número de estaciones y un modelo más complejo y extenso, aplicar un filtrado puede ser una buena opción para ahorrar tiempos de generación y obtener mallas mucho más optimizadas en algunas aplicaciones. Todo dependerá de la finalidad a la que vaya destinado nuestro proyecto. Los filtros para las nubes de puntos los podemos encontrar en el apartado “Filters” -> “Sampling” donde encontramos un total de 13 filtros que podemos aplicar.

El siguiente paso es realizar la reconstrucción de la malla mediante la nube de puntos. Para este caso en particular se han aplicado dos algoritmos distintos que son frecuentemente utilizados y se compararan los resultados de ambos, optando finalmente por el que genere mejores resultados para el fin del proyecto.

Los algoritmos de reconstrucción utilizados se encuentran en el apartado “Filters” -> “Remeshing, Simplification and Reconstruction” y los utilizados en este apartado han sido “Surface Reconstruction: Ball Pivoting” y “Surface Reconstruction: Poisson”.

En primer lugar analizaremos el algoritmo de Ball Pivoting. Este necesita del cálculo realizado anteriormente de las normales de la nube de puntos y su funcionamiento se resume en que éste comienza por un primer vértice de partida y a raíz de éste comienza a calcular el radio del próximo punto más cercano al anterior y crea una unión entre estos. El proceso continúa repitiéndose hasta completar el total de los puntos de la nube y va generando triángulos entre los vértices conectados hasta completar la superficie.

Básicamente con este algoritmo se consigue una superficie triangulada utilizando los tres vértices más cercanos.

Los parámetros utilizados para este método son los siguientes: (Fig. 6.25)

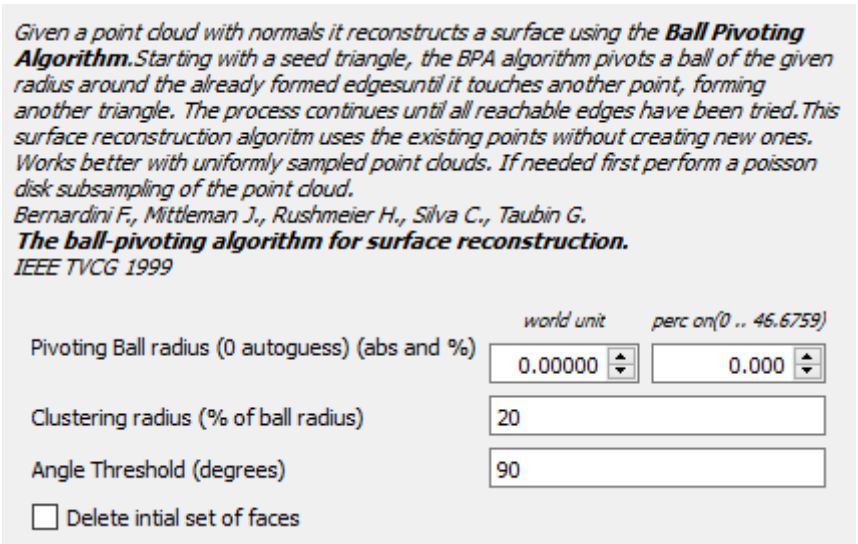


Figura 6.25. Algoritmo Ball Pivoting.

El resultado de aplicar este algoritmo es el que se muestra en (Fig. 6.26), (Fig. 6.27) y (Fig. 6.28).

Cabe destacar que la superficie generada es muy precisa ya que se trata de una triangulación entre los puntos captados por el escáner, pero la problemática que presenta es la gran cantidad de polígonos que se generan, dando como resultado un modelo muy detallado, pero a la vez muy pesado y difícil de gestionar. También cabe comentar que este algoritmo no es capaz de reconstruir las zonas de sombra que se generan con el escaneado, siendo necesario la realización de un mayor número de escaneados para cubrir estas zonas o bien una posterior reconstrucción de las mismas mediante procesos de modelado, como se puede observar en (Fig. 6.29).

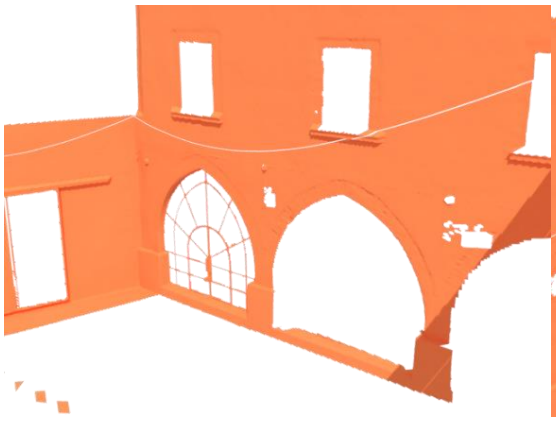


Figura 6.26. Superficie Ball Pivoting.

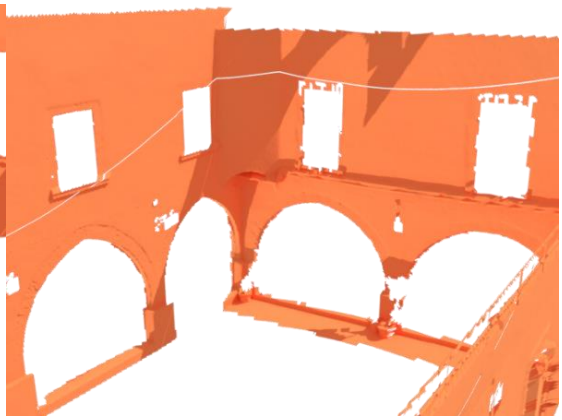


Figura 6.27. Superficie Ball Pivoting 2.

Figura 6.29. Polígonos Ball Pivoting.

Figura 6.27. Superficie Ball Pivoting 2.

El segundo algoritmo aplicado es el de Poisson, este consiste en una reconstrucción de la malla mediante la aproximación de una función escalar que mejor se ajuste a la superficie representada por las normales. Los parámetros utilizados se muestran en la (Fig. 6.30).

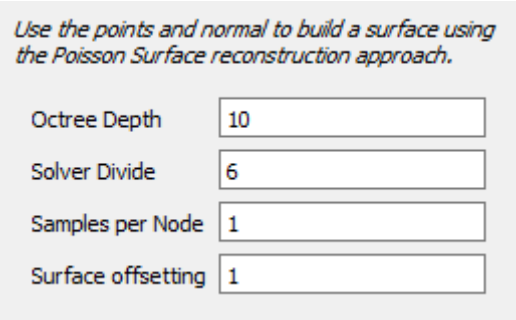


Figura 6.30. Algoritmo Poisson.

La superficie generada con Poisson nos da como resultado una malla menos pesada, que se ajusta a la superficie, pero que en los bordes rectos de la superficie genera formas más suaves como podemos apreciar en las imágenes, perdiendo ciertos detalles en algunas zonas, como las puertas y ventanas. Como punto a favor para este algoritmo cabe destacar que la superficie que se genera es continua y cerrada, por lo que no presenta agujeros (Fig. 6.31) y (Fig. 6.32).

Los dos métodos anteriores nos han permitido realizar la reconstrucción de la malla, pero ambos nos presentan la problemática de tener un elevado número de polígonos, como se aprecia en la (Fig. 6.32). Ésto representa un problema para este proyecto, ya que la finalidad es la exportación al motor gráfico de Unity. Además la estructura de las caras hace muy complicada la texturización del propio objeto de manera manual, cierto es que podemos utilizar de forma rápida las propias imágenes del escáner o el color de la nube de puntos, pero los mejores resultados se obtienen mediante la texturización UVW o con una parametrización y proyección de las fotografías en superficie. Para la texturización plana UVW necesitamos un modelo estructurado y con sus caras organizadas, mientras que si proyectáramos fotografías estas superficies nos serian completamente válidas.

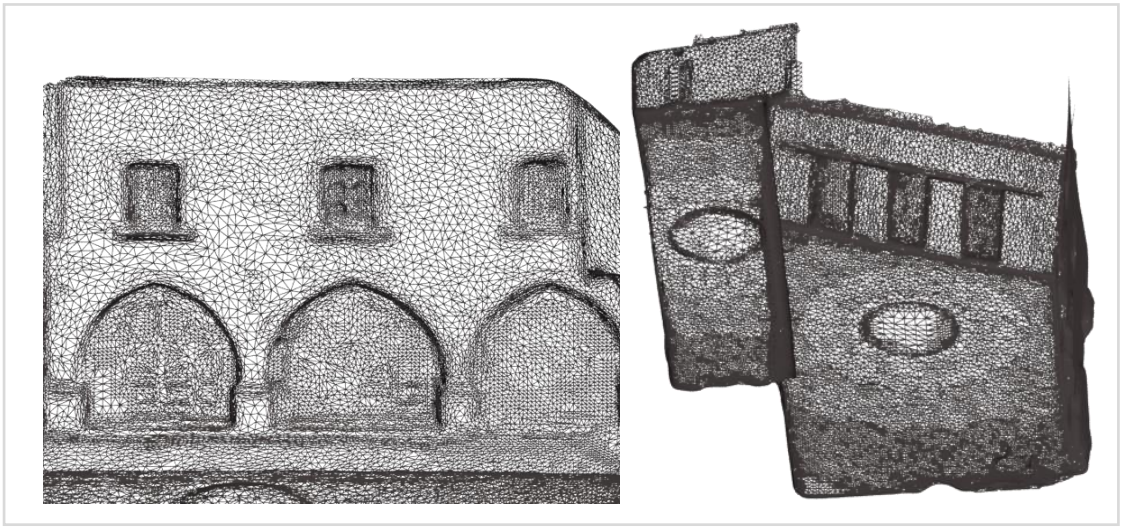


Figura 6.31. Superficie Poisson.

6.4.3. Retopología

Expuesto lo anterior y para optimizar el modelo para que resulte menos pesado, se ha optado por la realización de una retopología de la malla. Mediante éste proceso vamos a reconstruir de forma manual la superficie, ajustándola a nuestras propias necesidades.

La retopología del modelo pasa por utilizar una malla de referencia, que en nuestro caso es la procedente del escáner, para así generar una nueva superficie. Ésta se ajustará a la anterior mediante una tolerancia que especifiquemos. Como queremos mantener la máxima precisión en este proceso, simplemente debemos ajustar la tolerancia a 0. Así conseguimos que los vértices de los polígonos que añadidos se creen en la misma superficie del escaneado.

Las herramientas de retopología que vamos a utilizar se encuentran en el "Ribon" de modelado de 3Ds Studio Max. Simplemente debemos elegir el escaneado como superficie, crear un nuevo objeto e ir añadiendo vértices de los puntos clave que definen la geometría del objeto. Después se generarán las caras y se añadirán las subdivisiones necesarias. También podemos utilizar la herramienta "Conform" para ajustar a la malla las subdivisiones creadas.

En la (Fig. 6.32) se muestra el resultado de la retopología para el modelo de Poisson.

Como se aprecia en la (Fig. 6.32) las caras aparecen mucho más estructuradas y se resuelven las problemáticas anteriormente descritas.

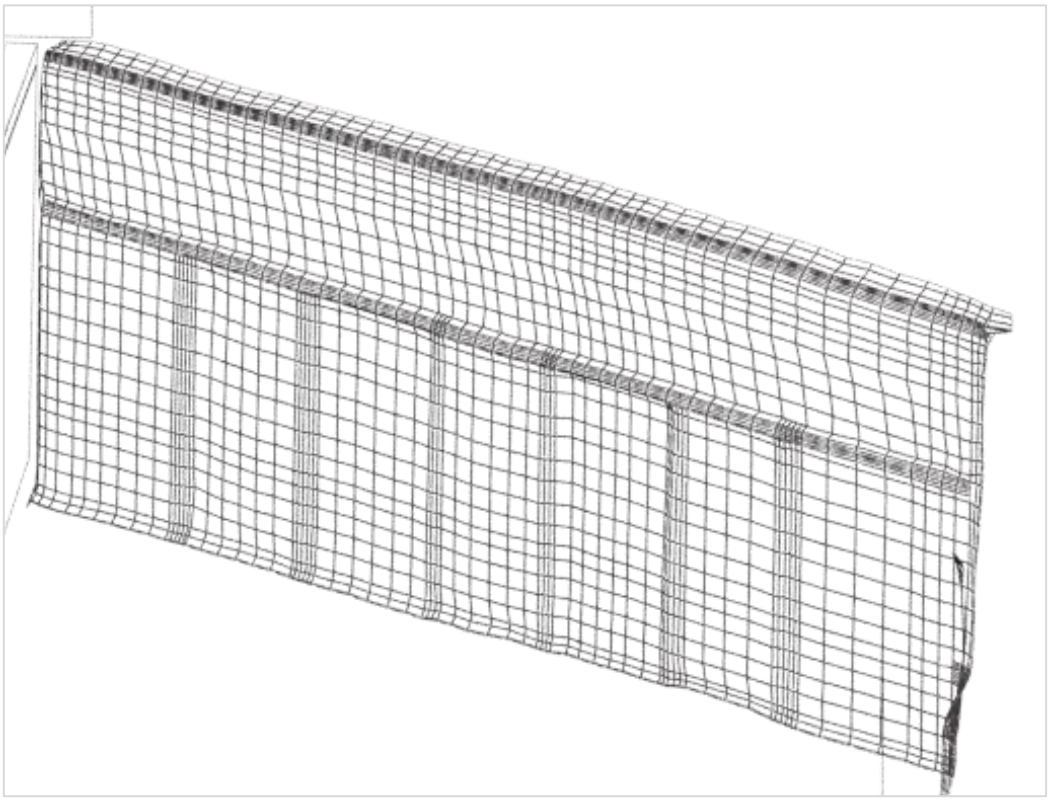


Figura 6.32. Retopología Poisson.

Por último, se muestra la retopología del modelo Ball Pivoting,(Fig.6.24) y (Fig. 6.25) que ha sido finalmente el utilizado para la reconstrucción del modelo escaneado, ya que como hemos comentado anteriormente, es el que mejor mantiene los detalles y no crea bordes suavizados en los objetos, resultando ser el modelo más realista.

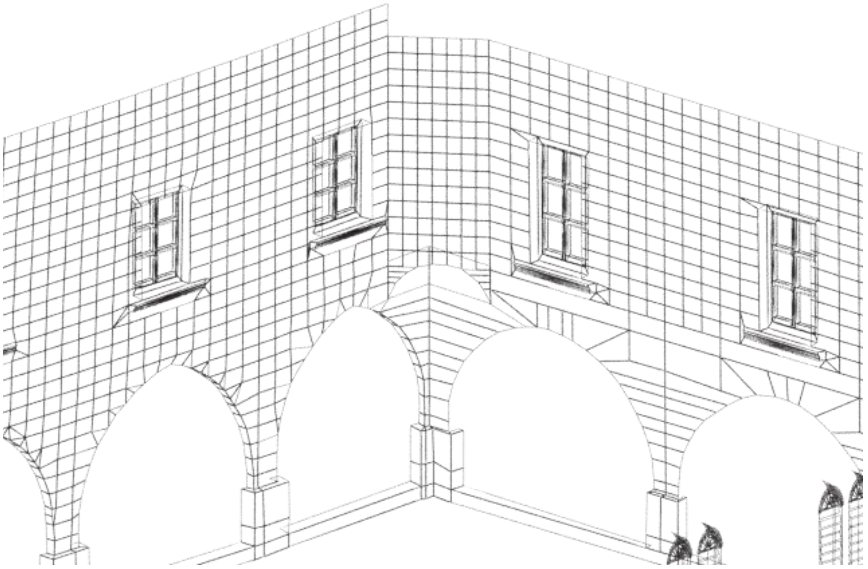


Figura 6.33. Retopología Ball Pivoting.

Figura 6.34. Retopología Ball Pivoting 2.

Gracias a esta metodología contamos con un modelo realista a los datos del escaneado y completamente optimizado. Pudiendo añadirle mayor o menor resolución en función de las propias necesidades de su uso. Finalmente el resultado final se muestra con un color de malla sólida en la (Fig. 6.26) y con la textura aplicada de las imágenes del láser escáner en (Fig. 6.27), (Fig. 6.28) y (Fig. 6.29). Nótese que la calidad fotográfica es mala debido a la captura de las imágenes con la cámara del equipo Leica ScanStation 2.

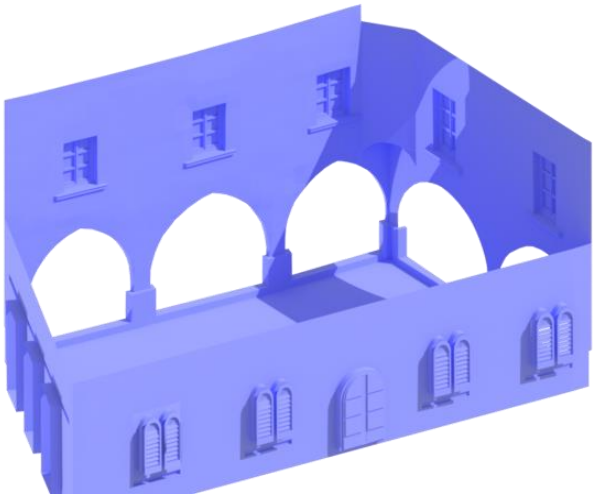


Figura 6.35. Retopología color sólido.

Figura 6.36. Retopología texturizado 1.

Figura 6.37. Retopología texturizado 2.

6.4.4. Comparación de mallas

Para poder cuantificar la precisión obtenida del modelo de retopología con el obtenido del escáner se ha utilizado el programa 3DReshaper. Para ello se han cargado las dos mallas citadas, utilizado el apartado de medición, seleccionado ambas y se han comparado.

En el gráfico (Fig. 6.30) se muestra que el 91.3% de la malla se corresponde con valores centrados de la distribución.

Los puntos que menor precisión presentan, se encuentran en las zonas de las ventanas. Como se aprecia, en la parte superior izquierda, se encuentran los peores resultados, ya que en este punto comienza la curvatura del torreón y en la retopología se ha omitido el torreón, dándole continuidad a la pared de forma recta.

Podemos afirmar que el resultado de la retopología de malla presenta un gran número de ventajas respecto a la utilización de las mallas utilizadas directamente del escáner. Presentan menor complejidad, menor número de polígonos y de tamaño de archivo, por lo que será mucho más óptima en cualquier entorno, tanto de diseño como de visualización.

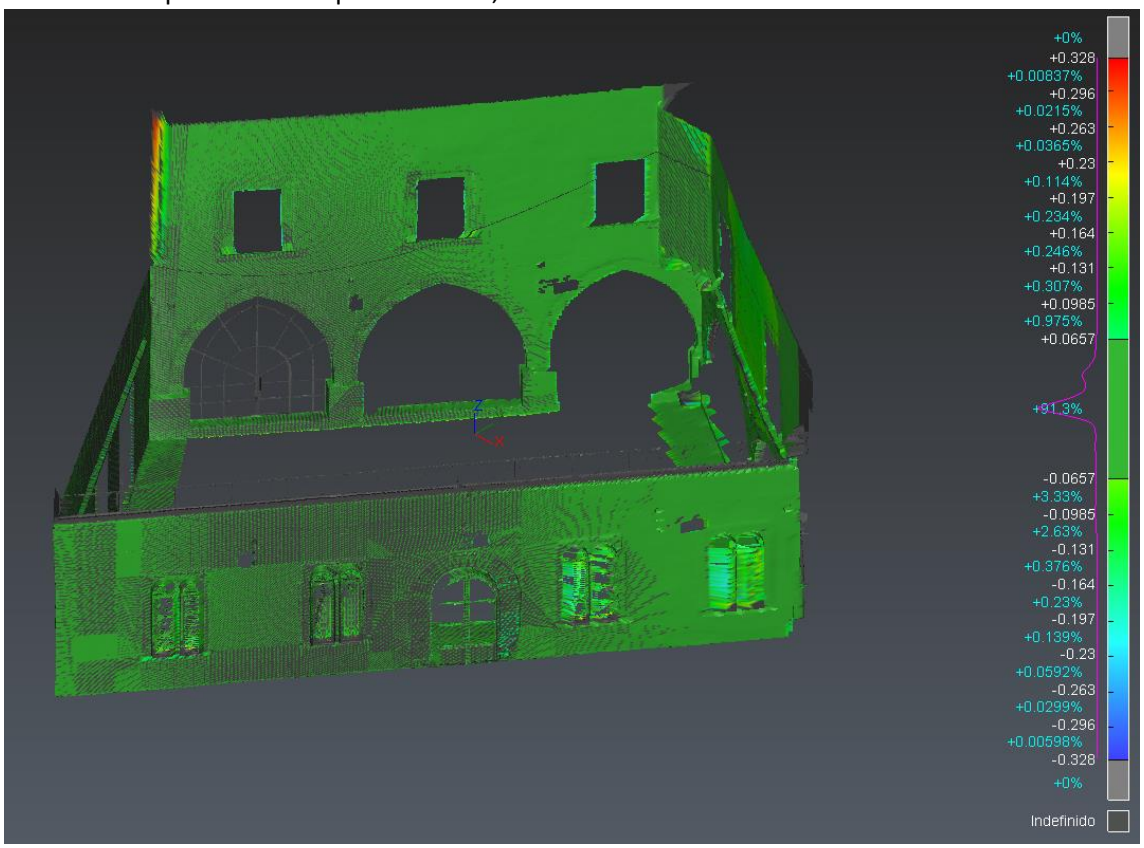


Figura 6.38. Comparación de mallas en 3DReshaper.

También se ha realizado una comparación de mallas con el programa MeshLab, para ello debemos primero debemos cargar las dos mallas que queremos analizar y después dirigirnos a la pestaña de “Filter” -> “Sampling” y seleccionar ”Hausdorff Distance”. Por último dentro de este apartado seleccionamos las dos mallas y esperaremos que termine el proceso.

La distancia de Hausdorff representa la distancia máxima entre ambas mallas (Fig. 6.31), por lo que no representa una medida de distancia real, pero permite comparar la calidad de la malla en los distintos puntos de la misma. El algoritmo de cálculo que utiliza MeshLab para esta tarea es el siguiente:

$$d_{\mathrm{H}}(X,Y) = \max\{ \sup_{x\in X} \inf_{y\in Y} d(x,y), \sup_{y\in Y} \inf_{x\in X} d(x,y) \}$$

Figura 6.39. Ecuación de Hausdorff (Fuente: MeshLab documentation manual).

En la (Fig. 6.32) podemos ver el resultado del algoritmo, como resultado muestra las distancias máximas y mínimas en pulgadas (in). Sitúa la media de la malla en el 0.084667 in, equivalente a 0.2150542 cm. Sitúa los valores entre 0 in y 1.129827 in, un resultado bastante bueno, que equivalen a 0 cm y 2.869761 cm. Como hemos visto anteriormente en la anterior comparación, la mayoría del modelo se ajusta al escaneado en valores próximos a 0.22 cm del intervalo de distancias.

```
Hausdorff Distance computed
Sampled 14103 pts (rng: 0) on ScanTopologia.obj
searched closest on Scanfiltrado.obj
min : 0.000000 max 1.129827 mean : 0.084667 RMS :
0.160462
Values w.r.t. BBox Diag (25.747902)
min : 0.000000 max 0.043880 mean : 0.003288 RMS :
0.006232
```

Figura 6.40. Algoritmo de Hausdorff.

Después aplicamos nuevamente otro filtro para la calidad de los puntos, para ello debemos dirigirnos a “Filters”-> “Color” y seleccionar “Colorize by quality”. Éste filtro nos proporcionara la calidad de los puntos de la comparación de las mallas, que va desde el rojo como mejor calidad, hasta el azul, como la peor. Lo que significa que si la malla presenta color rojo o tonos próximos a éste podremos afirmar que la calidad es buena, mientras que valores naranjas, verdes y azules representaran por este orden una peor calidad.

Por último aquí vemos el modelo de comparación generado por MeshLab (Fig. 6.32). Cabe destacar que se observan los mismos resultados que en la anterior comparación. Aunque en este caso la comparación se ha realizado de manera inversa. De la misma forma, observamos valores más altos en las ventanas superiores, con tonos amarillos y verdes. Ésto ocurre ya que en el modelo escaneado se han filtrado las ventanas y por lo tanto, no existe malla en estos puntos. Por ello obtenemos los valores más altos en estos puntos, pero si omitimos el análisis de estos puntos, los resultados son equivalentes.

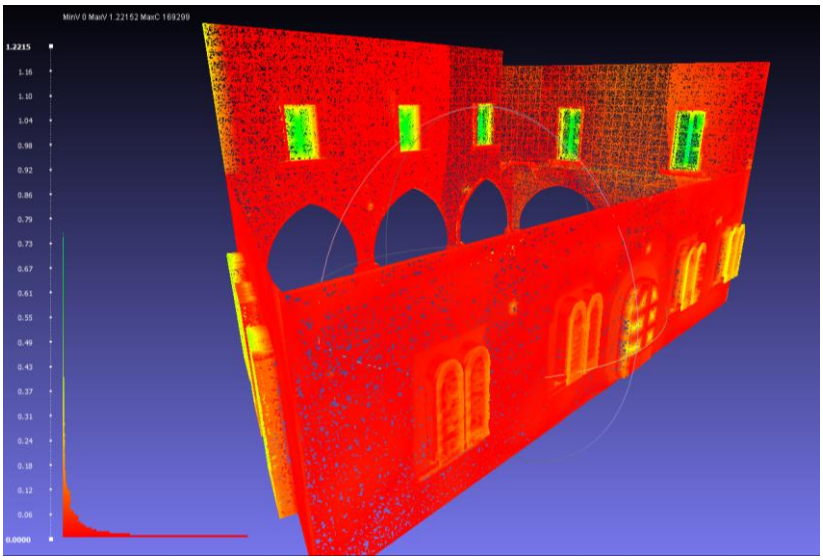


Figura 6.41. Comparación Hausdorff MeshLab.

7. Texturización del castillo

El proceso de texturización consiste en referenciar una imagen plana a la geometría de un objeto 3D. Cuando se crea un nuevo objeto o malla, este puede adoptar un color solido de forma predeterminada. Se puede modificar de manera fácil su color, opacidad y reflectividad como parámetros básicos, pero si queremos añadir la imagen al propio objeto debemos realizar el proceso de texturización, para asignar la imagen al mismo, dotando al objeto de mayor realismo.

En este apartado analizaremos los distintos métodos empleados de texturización aplicados en el presente proyecto.

7.1. Mapeado de coordenadas UVW

Esta técnica de mapeado consiste en asignar unas coordenadas planas a las distintas caras de un modelo complejo. Consiste en separar las diferentes partes del modelo de tal forma que las resultantes puedan estirarse y conformar una sección plana y así posarse sobre la imagen plana que deseamos que cubra el objeto.

Para aplicar el mapeado UVW lo recomendable es eliminar todas las coordenadas de mapeado y material del objeto. En primer lugar seleccionamos el objeto y nos dirigirnos al menú con forma de martillo “Utilities”. Dentro de este apartado presionar el botón “More…”, a continuación nos aparecerá una ventana emergente. Buscamos en el listado la opción de UVW Remove y se acepta. Ahora en la parte derecha del programa nos aparece una sección de parámetros con dos botones. Éstos son Remove: UVW y Materials. Al pulsarlos eliminamos los materiales y coordenadas de mapeado asociados al objeto.

El siguiente paso consiste en convertir el objeto a polígono editable. Ésta parte no es necesaria, pero si no trabajamos con un Edit Poly podemos tener problemas al seleccionar y despiezar las distintas partes y caras del objeto. Por ello y puesto que convertir el objeto a polígono editable se realiza de manera muy rápida y sencilla recomiendo siempre antes de comenzar con el mapeado la conversión. Se realiza seleccionando el objeto y haciendo click con el botón derecho del ratón y seleccionar convert to-> Editable Poly. También podemos simplemente seleccionar el objeto y aplicar el modificador Edit Poly de la lista de modificadores del panel “Modify” situado a la derecha del programa.

Seguidamente volvemos al listado de modificadores y buscamos “Unwrap UVW”. Al seleccionarlo se añadirá al objeto y aparecen nuevos comandos. Éstos los podemos utilizar para seleccionar las distintas partes en las que queremos dividir el objeto y la forma o proyección que se le puede asignar a las mismas. Lo recomendable es intentar siempre que las proyecciones de las partes sean planas para generar la menor distorsión posible en las texturas. También podemos utilizar el comando de Pelt Map, que intentara estirar las distintas caras de la sección, como si unas cuerdas estirasen de la misma para convertirla en un plano.

Una vez divididas todas las partes del objeto se utiliza el botón “Open UV Editor”. Esta nueva ventana nos permite posicionar las distintas secciones dentro de un espacio plano con coordenadas 2D. Con ella podemos acomodar, unir, escalar y rotar las diferentes partes hasta crear la plantilla de la textura.

Lo óptimo para generar un buen mapeado de coordenadas es intentar unir el mayor número de piezas posibles por sus costuras. De esta forma se crea continuidad en las aristas y la textura queda representada de manera correcta.

También es recomendable seleccionar todas las partes y utilizar el botón “Rescale Elements”. Con ello se consigue que todas las caras guarden su verdadera proporción. Aunque también puede darse el caso de que existan zonas menos visibles o que requieran de menor detalle. Podemos optar por no reescalar estas zonas para que ocupen un menor espacio dentro de la plantilla y así poder dar mayor espacio a otras zonas.

Una vez concluido el proceso nos dirigimos a la pestaña de “Tools” y seleccionamos “Render UVW Template”. Se abrirá una nueva ventana con la plantilla del mapeado para el objeto y guardaremos el archivo como imagen. La (Fig. 7.1) muestra una plantilla de un edificio utilizado en el modelo.

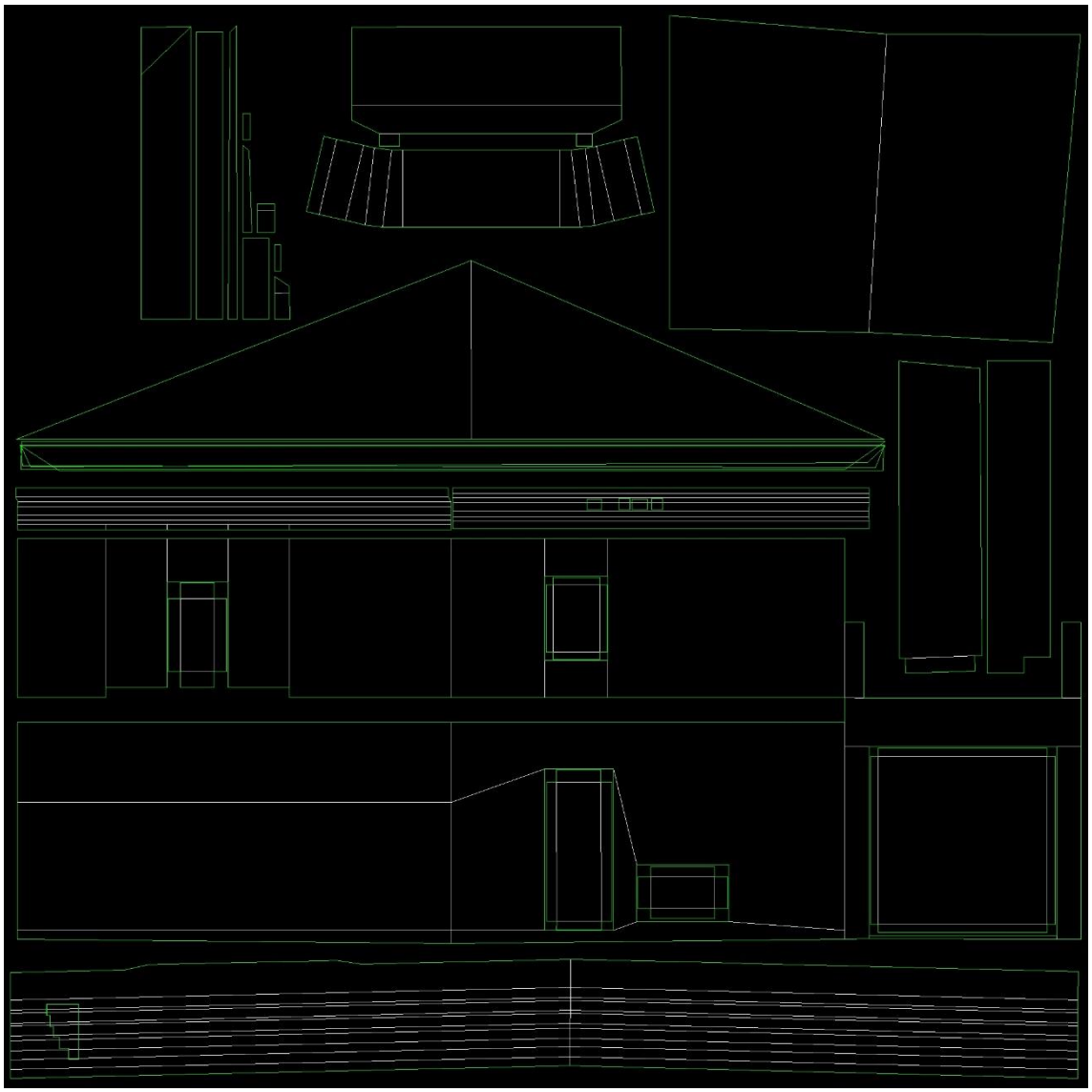


Figura 7.1. Plantilla textura edificio.

Ya generada la plantilla, ésta se abre con un programa de edición de imágenes. En este proyecto se ha trabajado con el programa Adobe Photoshop CS6, pero si deseamos utilizar software libre podemos realizar el mismo trabajo con Gimp de manera muy similar.

Dentro de Adobe Photoshop CS6 se carga la plantilla y procederemos a localizar las distintas imágenes correspondientes al objeto en cuestión. Realizamos la composición de la textura. Se aplican rectificaciones planas a las distintas partes que conforman la textura. Una vez terminada, se guarda en formato PNG con el mismo nombre del objeto. No es necesario que compartan el mismo nombre, pero sí muy recomendable, ya que se han generado un total de 98 texturas y trabajar de forma organizada nos permite localizarlas más rápidamente y tener un mayor control en este aspecto.

En la (Fig. 7.2) se muestra la textura terminada con la plantilla de la (Fig. 7.1).

Figura 7.2. Textura edificio.

Con la textura terminada, volvemos a 3D Studio y se aplica la textura sobre el objeto. Para ello se abre el editor de materiales, que se encuentra en la parte superior del programa o bien pulsaremos la letra “M”. Dentro de éste se crea un nuevo material y se le asigna el nombre del objeto en cuestión. Con el material ya renombrado nos dirigimos al canal difuso del material, se pulsa sobre éste y se selecciona la opción de “Bitmap”. Localizamos la textura en cuestión y se vincula, ahora el material ya tiene referenciada la textura.

Por último simplemente debemos asignar el material al objeto seleccionado, pulsar sobre el botón con forma de bombilla para habilitar la visualización en el espacio de trabajo y realizar un renderizado (Fig. 7.3) para comprobar que todo es correcto.

Figura 7.3. Renderizado edificio.

7.2. Mapas de Normales

Los mapas de normales permiten almacenar en cada pixel la dirección de la normal del objeto en cuestión. Éstos resultan muy interesantes, ya que añaden realismo a las texturas. Gracias a éstos, cuando se genere la iluminación, el motor de renderizado tendrá en cuenta las normales del objeto para proyectar la luz sobre el mismo, generando un mayor realismo y creando sensación de volumen.

Permiten trabajar con objetos de baja resolución y recrear los diferentes detalles del mismo sin tener caras adicionales que conformen los detalles.

Los mapas de normales pueden realizarse mediante una geometría más compleja ("High Poly") de mayor resolución y exportarla al objeto de menor resolución ("Low Poly").

Como ejemplo, se puede generar el mapa de normales de una pared de ladrillos a un plano. Disponiendo del modelo plano y de mayor resolución con los diferentes detalles de los ladrillos.

Para esta labor podemos utilizar plugins destinados a este propósito para 3Ds Max o herramientas como ZBrush. La generación de estos mapas por este método presenta la ventaja de generar unos mapas de normales basados en la propia geometría, de forma que resultan ser muy realistas y correctos. Por otro lado presentan la problemática de tener que modelar los detalles y tener dos modelos del objeto, el de baja resolución y el detallado. Sin embargo, para los modelos procedentes de escaneados resulta ser una muy buena solución. Permiten conservar la geometría de los detalles, sin tener que utilizar mallas más pesadas.

Por otro lado, los mapas de normales también se pueden generar mediante las propias fotografías o texturas. Resultando ser menos precisos y pueden presentar algunas formas incorrectas. Al calcular las normales mediante imágenes se debe controlar y experimentar con los distintos parámetros que ofrecen los programas que las generan. Ajustando parámetros evitaremos incurrir en errores, como que los ladrillos se hundan sobre la pared en lugar de sobresalir.

Existen plugins para Adobe Photoshop que permiten generarlos de manera automática. Programas como Substance Designer realizan la misma tarea y dan un mayor control sobre los ajustes de la generación del mapa.

Por último Unity permite convertir la textura del objeto a mapa de normales, siendo esta última la que menor control y calidad nos genera, pero también la más rápida de crearlos.

Para utilizarlas en el material simplemente nos dirigimos al material en 3Ds Max y se selecciona el canal "Normal map". Después "Bitmap" y se localiza la imagen de normales. También podemos realizar el mismo proceso en Unity, pero al realizase en 3Ds Max, al exportar el modelo a Unity este paso ya estará realizado.

Como ejemplo podemos ver en la (Fig. 7.4) a la izquierda la textura en cuestión aplicada al canal difuso, el mapa de normales en el centro y la combinación de ambos en la parte de la derecha.

Se puede observar otro ejemplo en la (Fig. 7.5) aplicado directamente sobre un objeto. En este caso en la parte izquierda vemos la combinacion de la textura y el mapa de normales, en el centro solamente la textura y en la parte derecha el mapa de normales.

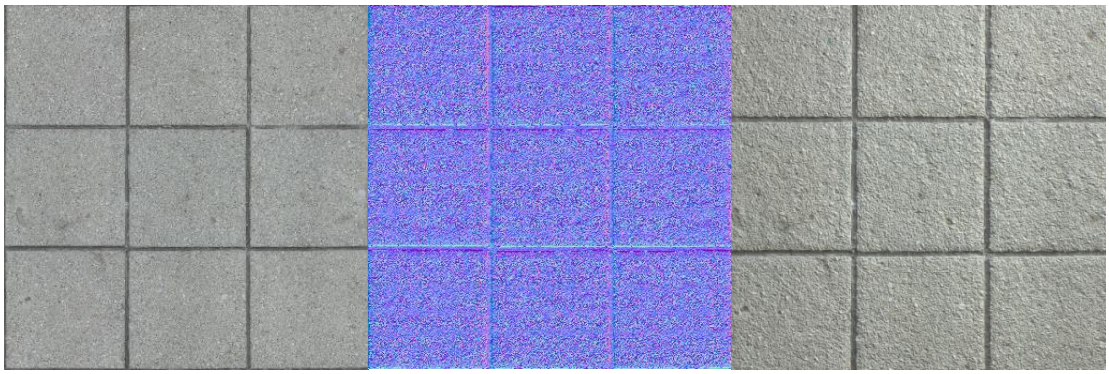


Figura 7.4. Mapas de normales.

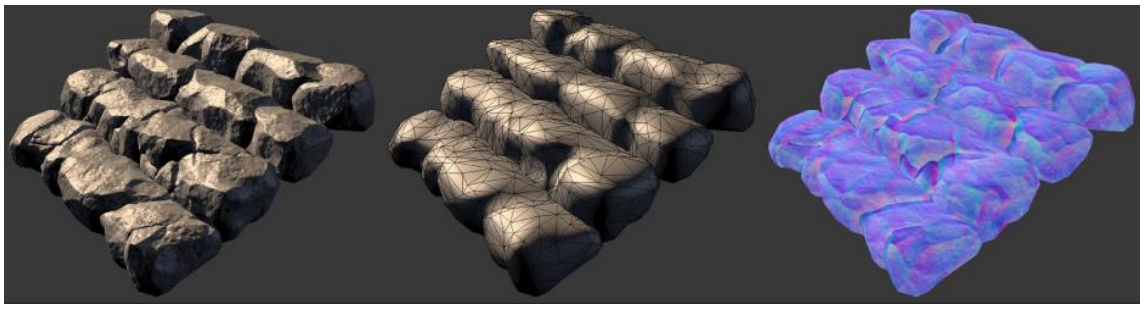


Figura 7.5. Mapas de normales 2 (Fuente: Polycount).

8. Preparación del entorno en Unity

Unity es un motor gráfico utilizado para hacer videojuegos que permite el renderizado en tiempo real de los diferentes objetos tridimensionales que añadamos al proyecto. Cuenta con físicas de objetos, efectos de iluminación y la posibilidad de escribir y ejecutar código.

8.1. Configuración del proyecto

En primer lugar debemos abrir el programa y elegir la opción de nuevo proyecto (Fig. 8.1). Se le asigna un nombre, la ubicación del mismo, si queremos que se optimice para elementos 3D o 2D y los paquetes de recursos que deseemos cargar.

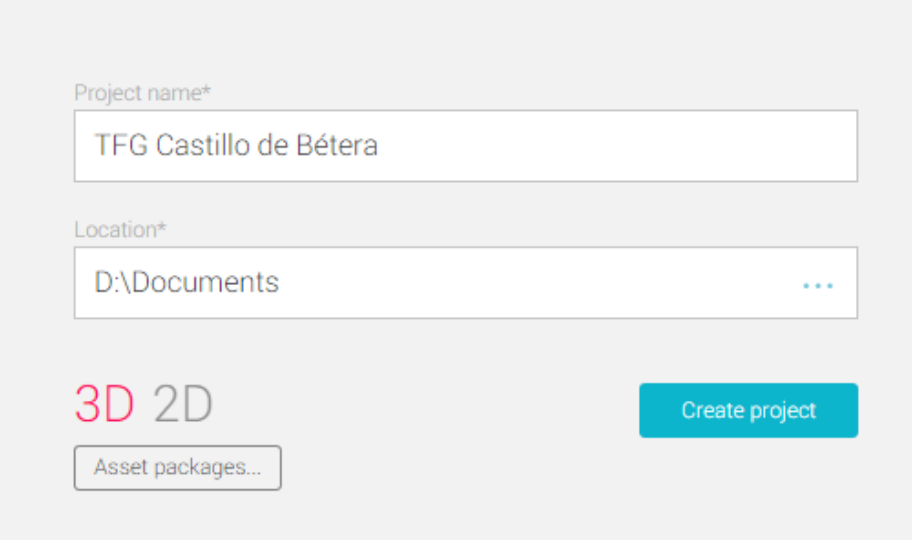


Figura 8.1. Nuevo proyecto Unity.

Hacemos click en el botón de “Asset packages” para seleccionar los recursos que nos resultaran útiles para empezar (Fig. 8.2). Podemos omitir esta parte y añadirlos más tarde, generar nuestros paquetes de recursos o descargarnos alguno que pueda resultarnos útil desde la Asset Store de Unity.

Para este proyecto se han cargado los siguientes recursos (Fig. 8.2):

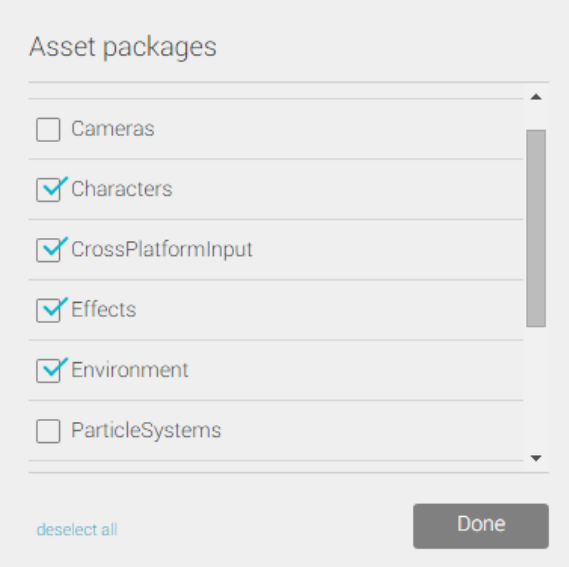


Figura 8.2. Assets Unity.

Cargamos Characters, que incorpora los controladores de primera y tercera persona. CrossPlatformInput, que contiene útiles para la exportación a dispositivos móviles. Effects que contiene scritps de cámara que utilizamos para dotar de mayor realismo y calidad al proyecto. Y Enviroment, que añade efectos de entorno.

Esperamos a que se cargue el proyecto, los distintos recursos y empezamos a configurar el proyecto.

Comenzaremos añadiendo el modelo 3D del castillo y sus texturas. Para ello, desde 3D Studio Max debemos exportar el modelo, bien a formato .FBX, .OBJ o simplemente utilizar el archivo en formato MAX.

El formato más recomendable es el .FBX, ya que presenta mayor compatibilidad.

Nos aseguramos que las unidades se encuentren en metros y que el eje Y se oriente hacia arriba (Fig. 8.3). Unity trabaja de manera correcta con estos parámetros. Si el modelo resulta estar girado en el momento de la importación a Unity siempre podemos rotarlo 90º grados sin mayor problema.

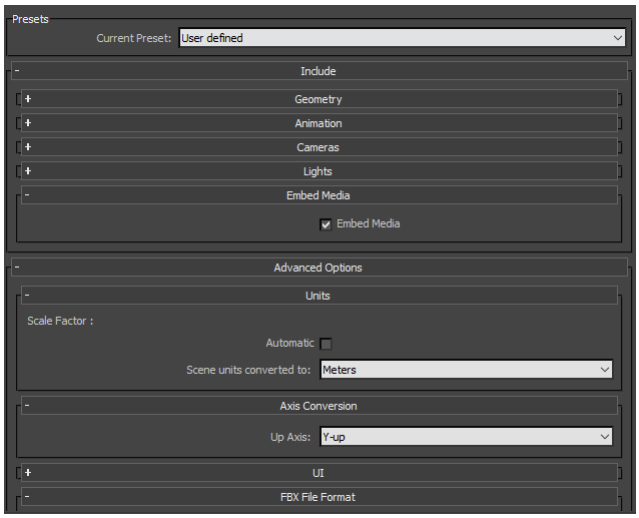


Figura 8.3. Exportación a FBX.

Creamos dos carpetas dentro de los Assets de Unity: “Modelos 3D” y “Texturas” (Fig. 8.4).

En primer lugar debemos arrastrar todas las texturas a la carpeta y después arrastrar el modelo en .FBX a la carpeta de modelos. Si se importa primero el modelo, Unity no dispondrá de las imágenes de las texturas y al importar los materiales del modelo no las encontrara, por lo que tendríamos que referenciar todas las imágenes a sus correspondientes materiales.

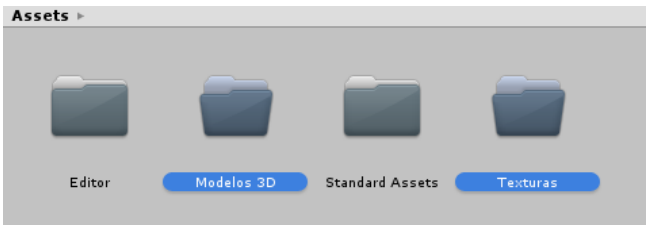


Figura 8.4. Carpetas Unity.

Esperamos a que se importen todas las texturas y el modelo. Después entramos en la carpeta de modelos, arrastramos el modelo a la parte izquierda del programa, donde pone Hierarchy (Fig. 8.5). Eliminamos el objeto Main Camera, ya esta se carga por defecto pero no es la cámara que utilizaremos.

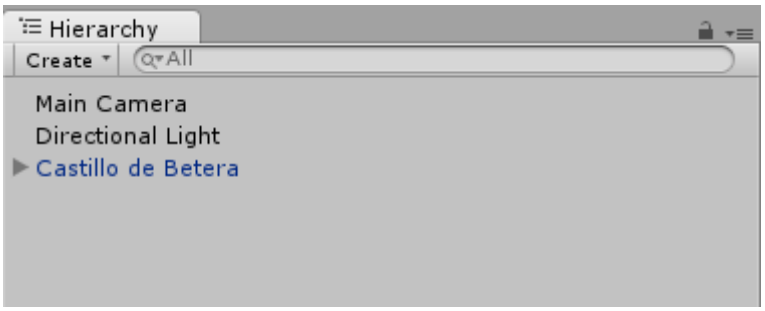


Figura 8.5. Hierarchy Unity.

Desplegamos el objeto del castillo y seleccionamos todos los objetos que lo componen. Seguidamente en la parte derecha del programa, llamada Inspector, pulsamos sobre el botón Add Component. Dentro de éste buscamos Mesh Collider (Fig. 5.6). De ésta forma hemos añadido a todos los objetos del modelo colisiones. Con ello se consigue que cuando el usuario se mueva por el escenario y entre en contacto con un objeto el programa interprete que es un sólido y se genere la colisión.

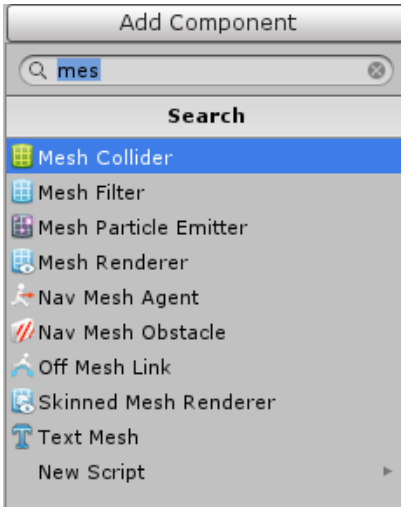


Figura 5.6. Mesh Collider.

Ahora nos dirigimos a la ruta que se ve en la (Fig. 8.7), se arrastra el componente FPSController dentro de Hierarchy y se escala hasta adaptarlo al modelo.

Figura 8.7. FPSController.

Dentro del objeto FPSController seleccionamos FirstPersonCharacter. En el inspector, se busca el apartado de cámara y se cambia el valor de Near de 0.3 a 0. Este parámetro controla la distancia de la cercanía de la cámara. Es necesario ajustarlo a 0 para que cuando nos acerquemos a un objeto la cámara no lo atraviese generando errores visuales.

Nos dirigimos a la ruta Assets->Estándar Assets->Effects->ImageEffects->Scripts y se arrastra dentro del inspector los siguientes scripts para la cámara (Fig. 8.8).

- Antialiasing: es el encargado de suavizar los dientes de sierra que generan las aristas de los objetos
- Bloom: genera efectos adicionales de iluminación
- Sun Shafts: permite que se cree el efecto de halo de luz que entran por las ventanas
- Camera Motion Blur: realiza desenfoque de movimiento en la cámara cuando esta se encuentra en movimiento
- Screen Space Ambien Obscurance y Ambien Occlusion: generan efectos adicionales de sombra en las aristas de las habitaciones.

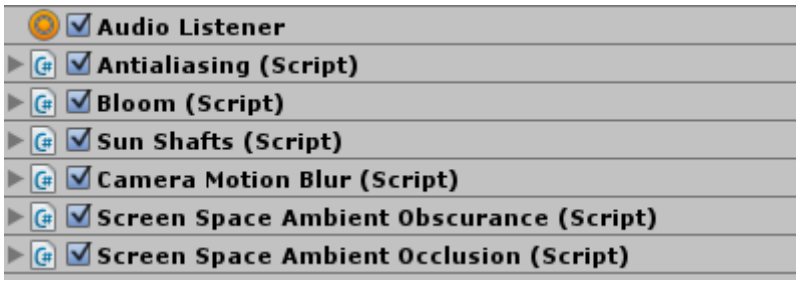


Figura 8.8. Efectos de imagen.

El siguiente paso será añadir el cielo y nubes para crear una representación más realista. Para ello nos dirigirnos a la parte superior del programa y seleccionar el apartado "Component"-> "Rendering" ->"Skybox".

Ahora nos aparecerá en el inspector un componente llamado SkyBox, dentro del mismo podemos seleccionar una textura de cielo y asignarla. Para este caso utilizamos el cielo que viene en el recurso de entorno que previamente hemos cargado, llamado Sunny Skybox.

A continuación nos dirigimos a FPSController. En el inspector, dentro del apartado “First Person Controller (Script)” ajustamos el parámetro de Jump Speed de 10 a 0. De ésta forma desactivaremos el salto del controlador, evitando que el usuario pueda saltar por el modelo y salirse del entorno.

También se ha tenido que evitar que el usuario pueda caerse desde lo alto de los torreones. Para ello la solución más práctica es añadir unas cajas a los bordes de los torreones (Fig. 8.9). Éstas serán las encargadas de que el controlador colisione y evite la caída al suelo. Simplemente las añadimos y posicionamos en los bordes de los torreones. Para que no se visualicen, nos dirigimos al inspector y desactivaremos el parámetro de Mesh Renderer para que no se visualicen.

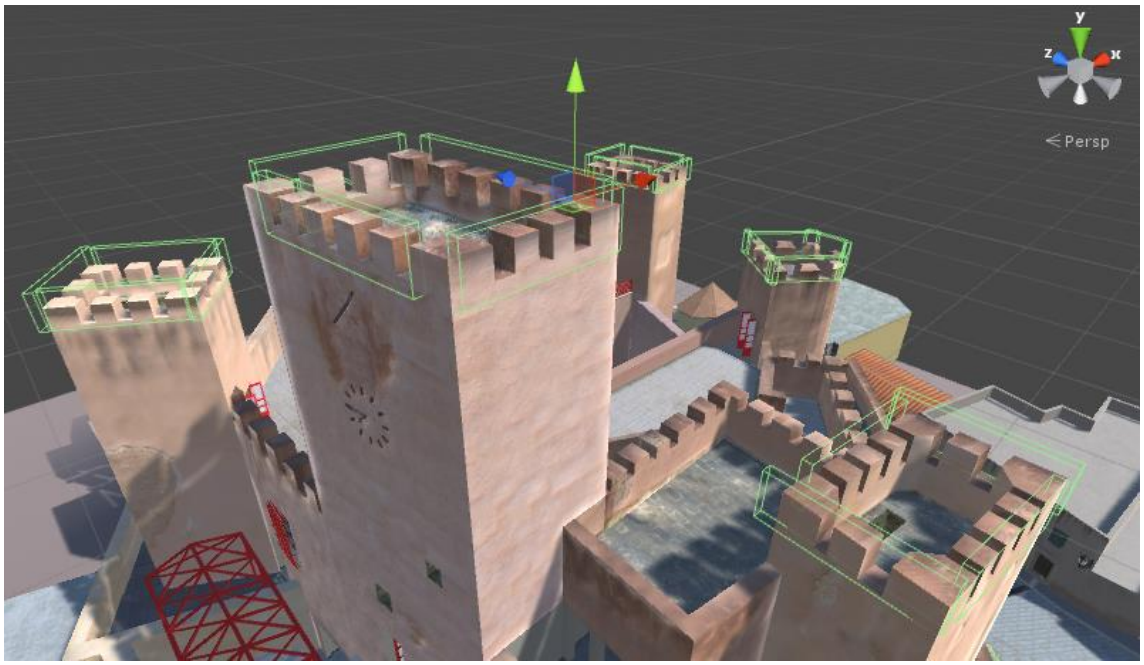


Figura 8.9. Colliders torreones.

El modelo cuenta con una estructura interior, exterior y además presenta una gran complejidad por las distintas plantas. Por ello se ha añadido un mini mapa, que permite posicionar al usuario dentro del modelo y facilitar la navegación sobre el mismo.

Para ello se ha añadido una nueva cámara a la escena. Ésta se posiciona exactamente en las mismas coordenadas (X; Z) que el FPSController. Respecto a la coordenada Y, ésta es indiferente, siempre que se encuentre por encima de todo el modelo.

Se ajusta en el inspector el parámetro de proyección de la cámara a ortográfica.

Creamos una textura con una flecha de navegación de color verde y la añadimos al proyecto. Ésta también debe posicionarse de la misma forma que la cámara anterior. Además se ha orientado la textura para que coincida con la dirección de la cámara del componente FPSController. También se ha añadido un plano en el cual se proyectarán las texturas con las imágenes del mini mapa.

Arrastramos los objetos dentro del componente de FPSController (Fig. 8.10). De ésta forma tanto la cámara, como la flecha de navegación heredan el movimiento del controlador.

Figura 8.10. Estructura del FPSController.

Como siguiente paso añadimos las texturas del minimapa al proyecto. Éstas irán cambiando en función de la posición en la que se encuentre el usuario. Se trata de siete planos que corresponden a las cuatro plantas del castillo, al entorno exterior, las escaleras de acceso traseras y a las terrazas del castillo.

Para conseguir el funcionamiento del mini mapa se crea la siguiente organización (Fig. 8.11). Los siete mapas se sitúan dentro del objeto Minimapas. También se sitúa el plano de proyección añadido previamente junto con sus respectivos activadores de cambio de textura también llamados Triggers.

Para los Triggers se han creado objetos tipo cubo en las distintas zonas de paso del modelo. Éstas se corresponden con las distintas zonas de los planos. En el inspector se ha activado el parámetro de "Is Trigger" y desactivado "Mesh Renderer".

Figura 8.11. Estructura Minimapas.

A los distintos Triggers se les ha añadido los distintos fragmentos de código (Fig. 8.12) creados en lenguaje JavaScript. Sustituyendo las variables Texture, ReplacementTex y originalTex por las distintas variables renombradas para cada zona.

cambio_texturas

```
var triggerTarget : GameObject; //model to change
var originalTex : Texture;
var replacementTex : Texture;
function OnTriggerEnter(other : Collider){
   triggerTarget.GetComponent.<Renderer>().material.mainTexture = replacementTex;
}
function OnTriggerExit(other : Collider){
   triggerTarget.GetComponent.<Renderer>().material.mainTexture = originalTex;
}
```

Figura 8.12. Script cambio de texturas minimapas.

Estos Scripts se han añadido a todos los correspondientes Triggers de las distintas zonas. En éstos se han asignado las diferentes imágenes (Fig. 8.13). Las variables a las cuales se asignan son Trigger Target, a la cual se le arrastra el objeto Plano_Minimapa_Tex. Y para Original Text y Replacement Text se les añaden las texturas correspondientes a cada zona.

Figura 8.13. Variables scripts minimapas.

Para concluir con este apartado se ha modificado los siguientes parámetros de la cámara encargada del minimapa con los siguientes valores:

- Culling Mask: Ajustamos el valor a Transparent FX. Asignamos también la flecha de navegación y el plano del minimapa a la capa Transparent FX. De esta forma la cámara solo mostrara los objetos que se encuentren en esa capa, de forma que no se renderizarán todos los demás elementos.
- Viewport Rect: Se han cambiado los valores de X, Y a 0.01. Ahora la posición de la cámara pasa a estar de la zona central de la pantalla a la esquina inferir izquierda. También cambiamos los valores de W a 0.3 y H a 0.4, modificando de esta manera el tamaño de la misma para que no ocupe toda la pantalla.
- Depth: Cambiamos el valor de 0 a 1, con ello la cámara del minimapa se situará en un nivel superior a la del controlador de movimiento.

8.2. Compilación del proyecto para distintas plataformas

La compilación del proyecto permite generar una aplicación con todos sus recursos empaquetados en una aplicación ejecutable. De forma que el usuario que desee visualizar el proyecto no tenga que instalar el propio editor de Unity para ejecutarlo.

Para este proyecto se ha compilado el modelo del castillo de Bétera para PC, Navegadores Web y dispositivos Android.

A continuación se explican los distintos procedimientos para cada plataforma

8.2.1. Windows, Mac y Linux

La compilación del proyecto para ordenadores se puede realizar tanto para Windows, Mac y Linux. Ésta se realiza de la misma forma para las distintas plataformas, pero únicamente se ha compilado para Windows.

En primer lugar nos dirigimos a la pestaña “File” -> “Build Settings”.

En esta ventana (Fig. 8.14) seleccionamos la plataforma correspondiente y presionamos el boton “Switch Platform”. Esperamos a que el programa realice el cambio de plataforma y cuando termine la barra de progreso pulsamos sobre el boton “Add Scene”. Se presiona “Add Open Scenes” y seleccionamos la escena guardada del castillo. En Target Platform utilizamos la opción de Windows, se selecciona Architecture x86 para procesadores de 32 bits y x86_64 para procesadores de 64 bits.

Nos dirigimos al boton “Players Settings” y configuramos algunas opciones adicionales. Dentro de este apartado se permitir elegir la calidad, resolucion y si deseamos que se ejecute a pantalla completa o en ventana.

Añadimos en “Icon” la miniatura del archivo EXE, aceptamos y pulsamos el botón de Build. Seleccionamos la ruta en la que deseamos guardar la aplicación y esperemos a que finalice el proceso. Con todo esto ya tendremos la aplicación compilada como un ejecutable para ordenadores.

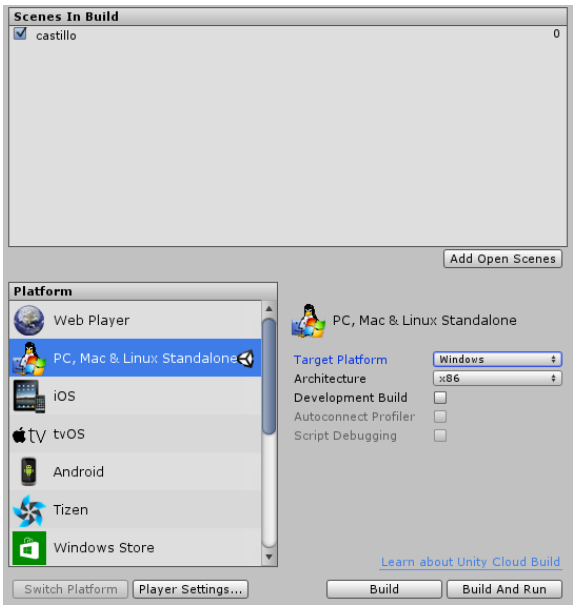


Figura 8.14. Build Settings.

8.2.2. Navegadores Web HTML

Para la compilación en navegadores web el proceso es muy similar al descrito anteriormente. Simplemente debemos dirigirnos al apartado de “Build Settings” (Fig. 8.14) como en el caso anterior. Dentro del mismo ir a “Player Settings”, en el apartado de resolución ajustar el tamaño que deseamos y marcar la casilla de no contex menú en el apartado de template.

En este caso se nos generara un archivo HTML que podemos cargar en nuestra página web. Para que se ejecute simplemente se requiere de un plugin para el navegador llamado Unity Player. Al momento de abrir la página el navegador nos dirá si queremos añadirlo y con todo esto ya funcionara. Podemos ver su funcionamiento en la (Fig. 8.15), ejecutándose desde el navegador Mozilla Firefox.

Cabe destacar que únicamente es compatible con los navegadores Mozilla Firefox, Internet Explorer y Opera, ya que Google Chrome en su última versión no permite la instalación del plugin de Unity, por lo que no será compatible con este último navegador.

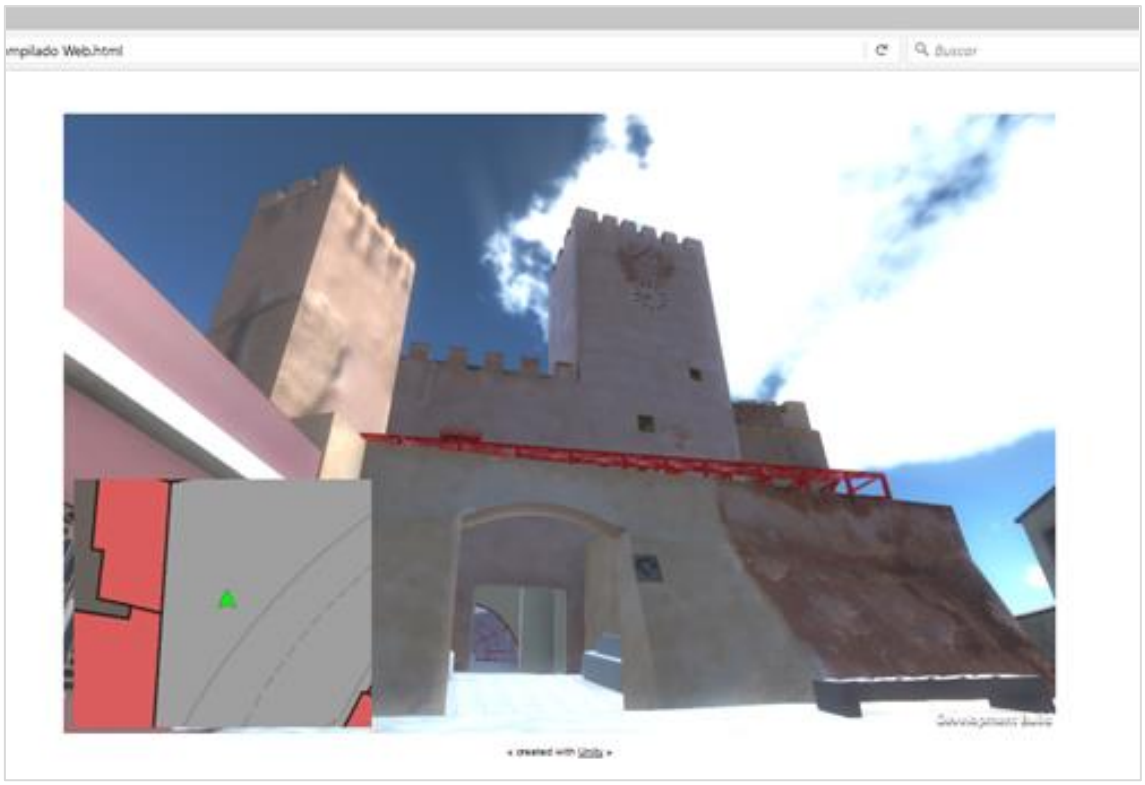


Figura 8.15. Proyecto compilado HTML5.

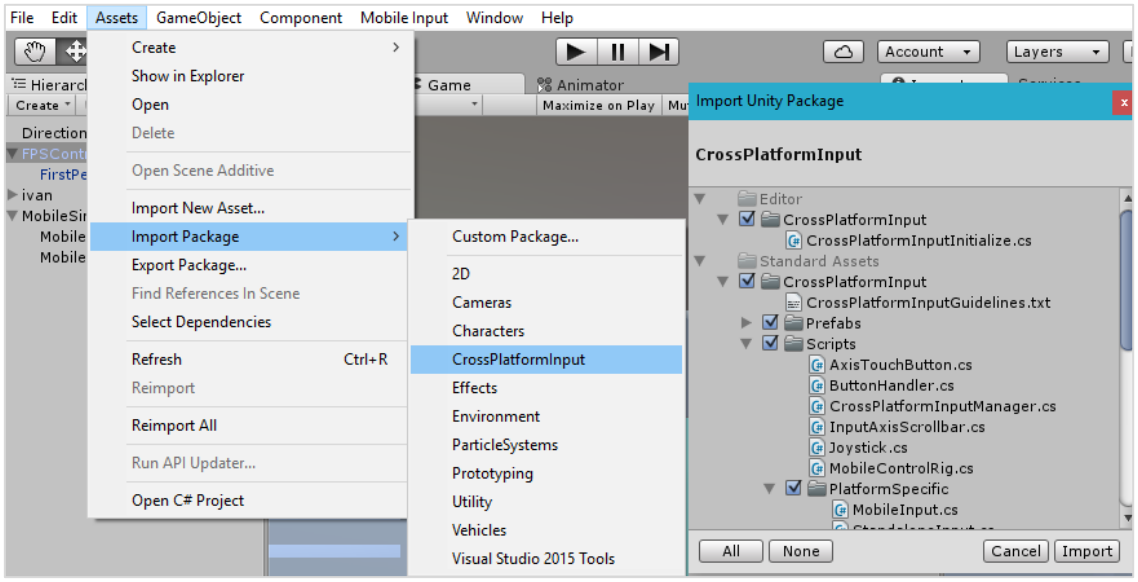
8.2.3. Dispositivos móviles con Android

La compilación para dispositivos Android es más compleja que en los casos anteriores. Se debe realizar una serie de pasos e instalar ciertos programas y librerías para poder generar el archivo .apk y así poder instalarlo en dispositivos móviles Android.

8.2.3.1. Añadir y configurar los controles táctiles

En primer lugar hay que añadir y configurar los controles móviles. Para ello tenemos que añadir los componentes First Person Controler y Mobile Single Stick Control.

En el caso de no haber importado el paquete movil, éste lo podemos encontrar en la pestaña de Assets, importar paquete y seleccionamos “CrossPlatformInput”. Se nos abrirá una nueva ventana emergente, en la cual debemos cercionarnos que esten marcados como seleccionados la carpeta de “Prefabs” y “Scripts” (Fig. 8.16).



2Figura 8.16. CrossPlatformInput.

Ahora simplemente debemos ir a la ruta Assets\Standard Assets\CrossPlatformInput. A continuacion nos dirigimos a Prefabs, seleccionamos el recurso de MobileSingleStickControl y se arrastra a la escena (Fig. 8.17).

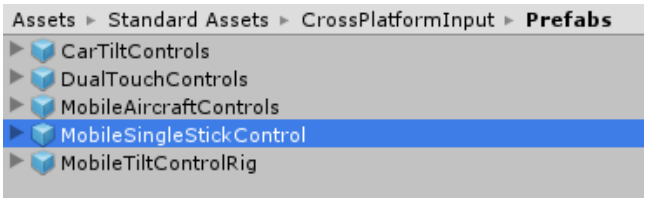


Figura 8.17. MobileSingleStickControl.

Hay que duplicar el componente MobileJoystick y renombrarlo. En este caso se ha cambiado el nombre por MobileJoystick Camera, que debe quedar de la siguiente forma (Fig. 8.18).

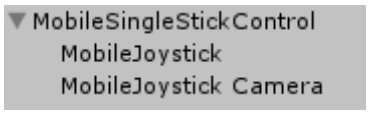


Figura 8.18. MobileJoystick.

Se elimina de “MobileJoystick Camera“ el script que viene integrado. Éste se llama “Joystick”. Añadimos el script llamado “TouchPad”. Éste se encuentra dentro de la ruta que podemos ver en la (Fig. 8.19).

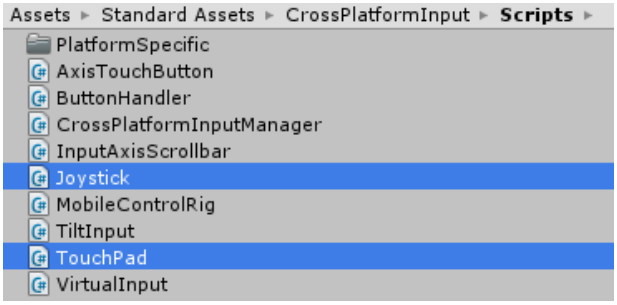


Figura 8.19. TouchPad.

Cambiamos los parámetros de posición del control y las variables de los ejes que se van a mover (Fig. 8.20).

Para ello se modifica el campo “Pos X” a un valor distinto para el otro control. Si no se realiza los dos se encontrarán superpuestos en la misma posición (Fig. 8.21). Tambien tenemos que cambiar el valor de Horizontal Axis Name por “Mouse X” y Vertical Axis Name por “Mouse Y”, el resto de parámetros podemos dejarlos como vienen por defecto.

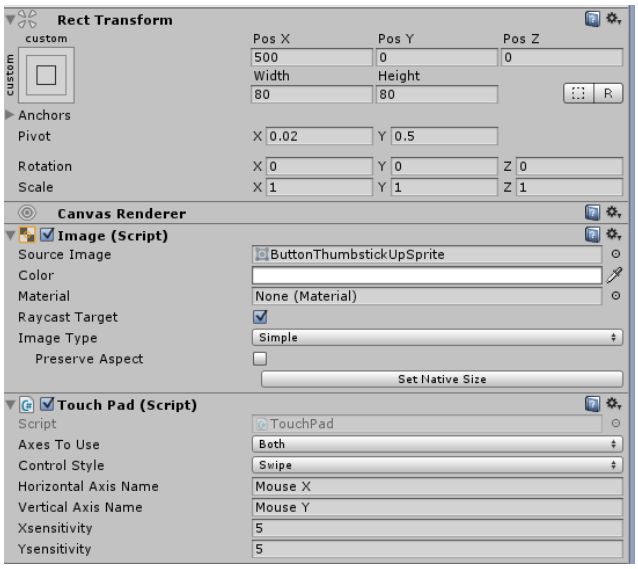


Figura 8.20. Pos X PosY.

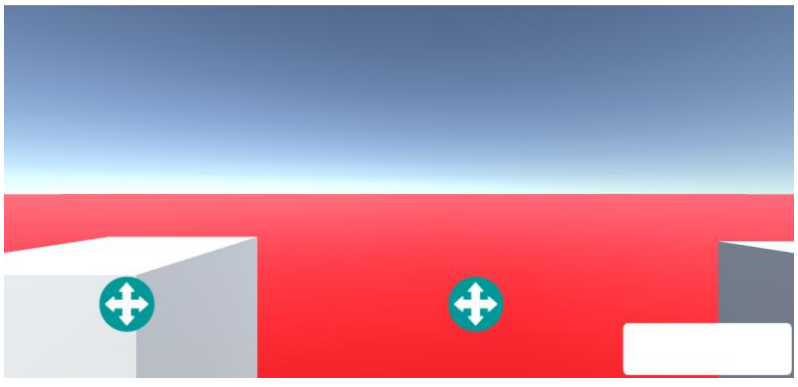


Figura 8.21. Posición Joysticks.

A continuación activamos “Enable mobile imput” y creamos un nuevo EventSystem que se encuentra en (Component->UI->Event System).

Abrimos “mobileSingleStickControl”, buscamos el script que tiene añadido de Joystick dentro del inspector y lo modificamos. Ésto sirve para corregir el error de la posición del mismo, ya que si no se modifica el controlador no regresa a su posición inicial. Por ello cambiamos la línea 28 “void OnEnabled”, por “void Start”.

En la siguiente imagen podemos visualizar el código utilizado para los controladores de movimiento (Fig. 8.22).

```
1  using System;
2  using UnityEngine;
3  using UnityEngine.EventSystems;
4
5  namespace UnityStandardAssets.CrossPlatformInput
6  {
7      public class Joystick : MonoBehaviour, IPointerDownHandler, IPointerUpHandler, IDragI
8      {
9          public enum AxisOption
10         {
11             // Options for which axes to use
12             Both, // Use both
13             OnlyHorizontal, // Only horizontal
14             OnlyVertical // Only vertical
15         }
16
17         public int MovementRange = 100;
18         public AxisOption axesToUse = AxisOption.Both; // The options for the axes that
19         public string horizontalAxisName = "Horizontal"; // The name given to the horizo
20         public string verticalAxisName = "Vertical"; // The name given to the vertical a
21
22         Vector3 m_StartPos;
23         bool m_UseX; // Toggle for using the x axis
24         bool m_UseY; // Toggle for using the Y axis
25         CrossPlatformInputManager.VirtualAxis m_HorizontalVirtualAxis; // Reference to t
26         CrossPlatformInputManager.VirtualAxis m_VerticalVirtualAxis; // Reference to the
27
28         void Start()
29         {
30             m_StartPos = transform.position;
31             CreateVirtualAxes();
32         }
33
34         void UpdateVirtualAxes(Vector3 value)
35         {
36             var delta = m_StartPos - value;
37             delta.y = -delta.y;
38             delta /= MovementRange;
39             if (m_UseX)
40             {
41                 m_HorizontalVirtualAxis.Update(-delta.x);
42             }
43
44             if (m_UseY)
45             {
46                 m_VerticalVirtualAxis.Update(delta.y);
47             }
48         }
49
```

Figura 8.22. Código controladores.

Renombramos el joystick derecho, nos dirigimos al inspector y los ejes del script (Horizontal por HorizontalLook y Vertical por VerticalLook) (Fig. 8.23).

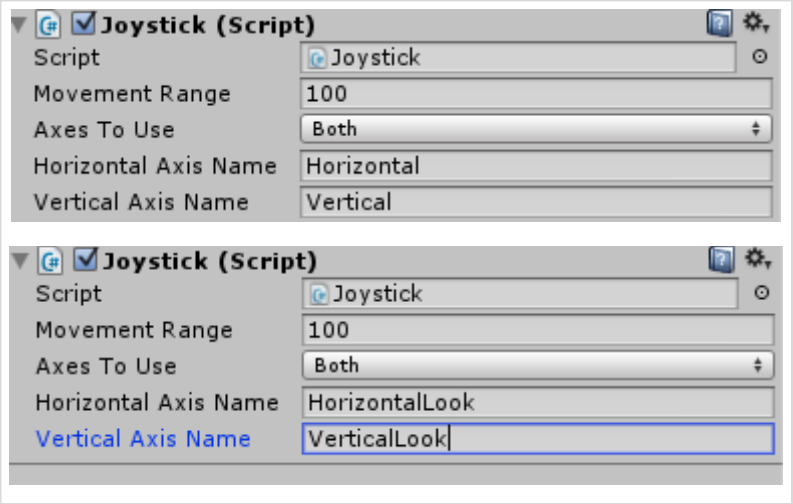


Figura 8.23. Ejes Joysticks.

Después modificamos el script FirstPersonController (Fig. 8.24).

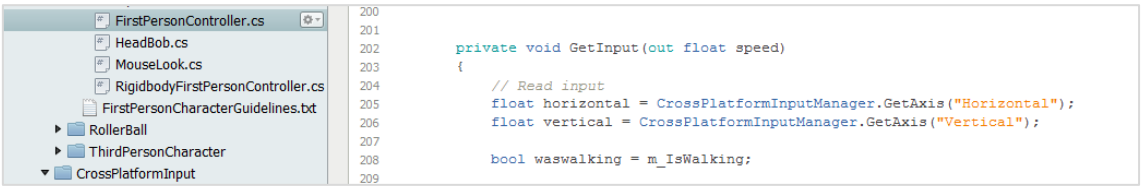


Figura 8.24. Script FirstPersonController Mobile.

Y ahora cambiamos las dos líneas de código de los ejes por las siguientes (Fig. 8.25):

(GetAxis -> GetAxisRaw)

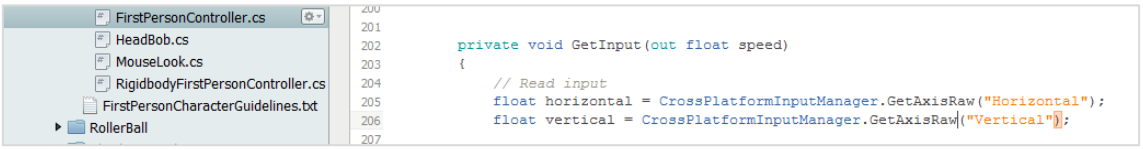


Figura 8.25. Script FirstPersonController Mobile modificado.

En éste momento ya tenemos configurados nuestros Joysticks. Éstos nos permitirán desplazarnos por el modelo y controlar la cámara.

8.2.3.2. Instalación de Java Se Development Kit

Para poder compilar la aplicación para Android en primer lugar hay que instalar java SE Development Kit.

Para saber si está instalado debemos buscar en la ruta "C:\Archivos de programa\Java\jre".

En la siguiente dirección podemos encontrar el link de descarga:

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Seleccionamos la versión para nuestro sistema operativo, que en nuestro caso es la versión de Windows de 64 bits (Fig. 8.26).

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	77.73 MB	jdk-8u73-linux-arm32-vfp-hflt.tar.gz
Linux ARM v6/v7 Hard Float ABI	74.68 MB	jdk-8u73-linux-arm64-vfp-hflt.tar.gz
Linux x86	154.75 MB	jdk-8u73-linux-i586.rpm
Linux x86	174.91 MB	jdk-8u73-linux-i586.tar.gz
Linux x64	152.73 MB	jdk-8u73-linux-x64.rpm
Linux x64	172.91 MB	jdk-8u73-linux-x64.tar.gz
Mac OS X x64	227.25 MB	jdk-8u73-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.7 MB	jdk-8u73-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.08 MB	jdk-8u73-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.36 MB	jdk-8u73-solaris-x64.tar.Z
Solaris x64	96.78 MB	jdk-8u73-solaris-x64.tar.gz
Windows x86	181.5 MB	jdk-8u73-windows-i586.exe
Windows x64	186.84 MB	jdk-8u73-windows-x64.exe

Figura 8.26. JSDK (Fuente: Java Oracle).

8.2.3.3. Android Studio

El siguiente paso es opcional pero recomendable. Se trata de instalar Android Studio (Fig. 8.27) lo podemos encontrar en el siguiente enlace de descarga:
http://developer.android.com/sdk/index.html

Aunque su instalación es opcional añade opciones interesantes como las máquinas virtuales. Además también nos permite instalar el paquete de Android SDK, aunque podríamos instalarlo por separado. Éstas son algunas utilidades que nos resultaran útiles para testear nuestra aplicación. Por ello en la práctica se recomienda su instalación.

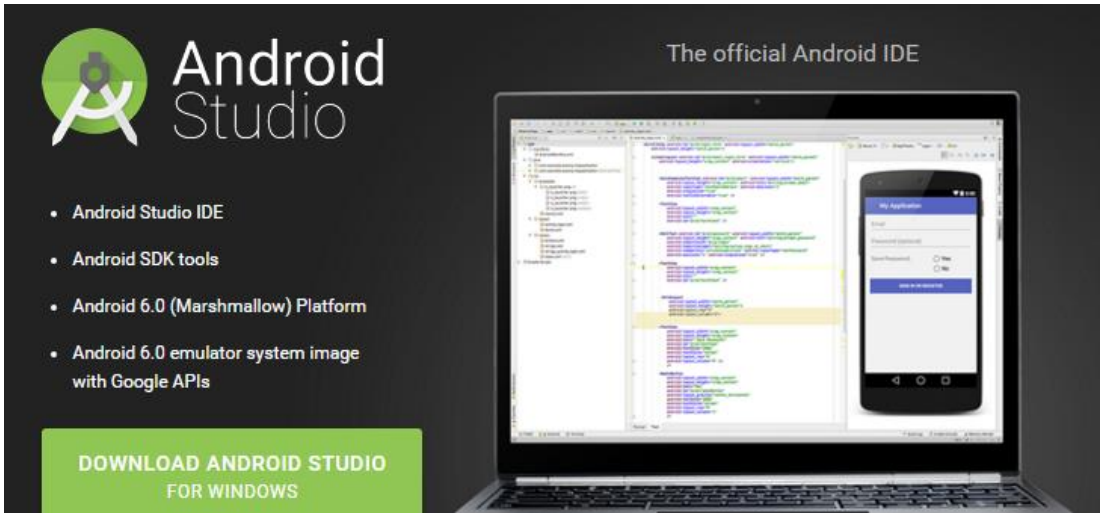


Figura 8.27. Android Studio (Fuente: Developer Android).

8.2.3.4. Android SDK

A continuación hay que instalar Android SDK. Este paquete es necesario para desarrollar cualquier aplicación en Android. Contiene los diferentes paquetes de librerías y herramientas para las distintas versiones de Android.

Lo primero que debemos hacer es descomprimir el paquete y seleccionar una ruta para el mismo. Aunque podemos tenerlo siempre que tengamos marcado la opción de incluirlo con la instalación de Android Studio.

Una vez descargado y localizado en nuestro disco duro, buscamos la ruta del mismo y abrimos la aplicación SDK Manager que encontramos. Ésta es la encargada de actualizar todos los recursos y paquete de las distintas versiones del sistema operativo de Android.

Para el correcto funcionamiento de nuestra aplicación en Unity debemos verificar en SDK Manager que tenemos instalados ciertos paquetes. En caso contrario procederemos a instalarlos mediante esta aplicación.

Los paquetes necesarios son:

- Android SDK Tools
- Android SDK Platform-tools
- La versión de desarrollo de correspondiente a la versión de Android que vayamos a utilizar (Android 4.0.3-API 15)
- Android Support Library
- Google USB Driver
- Intel x86 Emulator Accelerator (Haxm installer)

A continuación deberemos conectar nuestro teléfono por cable USB al ordenador. Esperar a que instale los driver el sistema operativo y situarnos en la carpeta de Android SDK. Pulsamos Shift (Izquierdo) y click derecho sobre la carpeta “platform-tools” y seleccionamos “Abrir ventana de comandos aquí” (Fig. 8.28). Se abrirá la terminal de Windows con la particularidad de que ya tendrá definida la ruta de la carpeta que hemos seleccionado (Fig. 8.29).

Una vez aquí escribimos *“ADB devices”* y comprobamos que aparezca nuestro dispositivo en la lista. Si no aparece o aparece como desautorizado debemos activar las opciones de desarrollador desde nuestro teléfono y activar el modo de depuración USB.

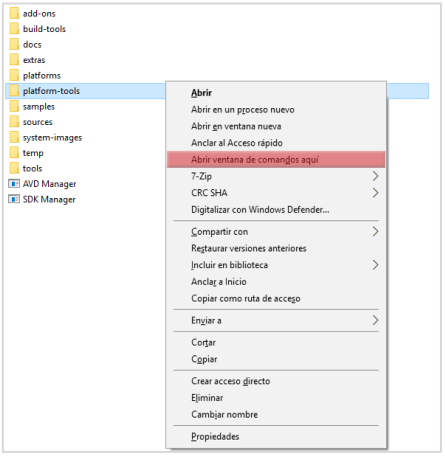


Figura 8.28. Ventana de comandos.

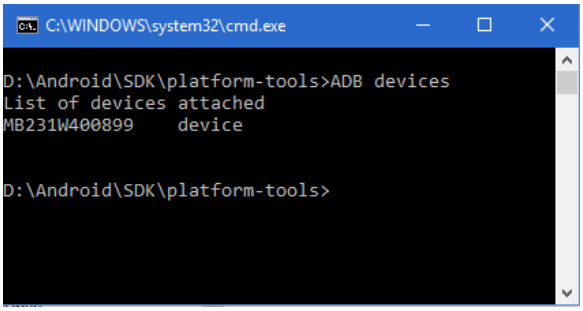


Figura 8.29. Terminal Windows.

8.2.3.5. Unity Remote 4

El siguiente paso consiste en instalar Unity Remote 4 en nuestro teléfono Android.

Unity Remote 4 (Fig. 8.30) nos permite ejecutar la aplicación directamente desde el móvil. Podremos realizar ésto si éste se encuentra conectado por USB y se ha configurado todo lo anterior de manera correcta. Su instalación es opcional, pero permite probar resultados en el dispositivo móvil sin tener que compilar la aplicación, generar el instalador (Apk), copiarlo al móvil e instalarlo. Por lo tanto también se recomienda su instalación. Podemos encontrar Unity Remote en el siguiente enlace:

https://play.google.com/store/apps/details?id=com.unity3d.genericremote&hl=es

Figura 8.30. Unity Remote (Fuente: Unity 3D).

8.2.3.6.Últimos ajustes de configuración en Unity

Ahora es el momento de Ajustar ciertos parámetros de Unity para trabajar con Android. El próximo paso a realizar consiste en decirle a Unity cuáles son las carpetas en las que tenemos instalado JDK y Android SDK.

Para ello seleccionamos la opción de preferencias que se encuentra dentro del apartado de Edición (Fig. 8.31). Después dentro de ésta seleccionamos “Herramientas Externas” y definimos las rutas.

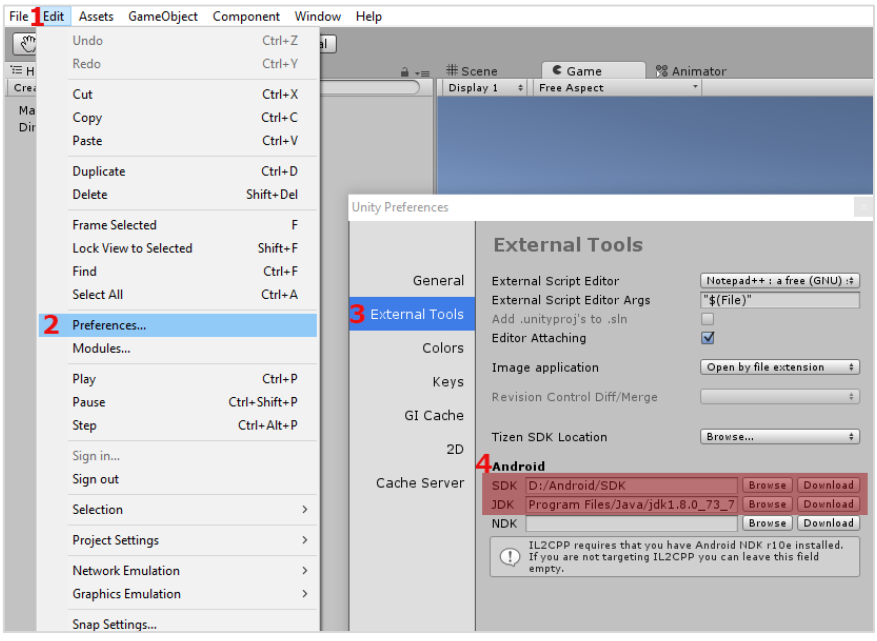


Figura 8.31. Rutas JDK y SDK en Unity.

Luego nos dirigimos a Edición-> Ajustes del Proyecto-> Editor.

Dentro de los Ajustes del Editor, en el apartado de Unity Remote, concretamente en la casilla de dispositivo se cambia la opción de “ninguno”. Ésta se encuentra marcada por defecto y la cambiamos a “cualquier dispositivo Android” en la pestaña desplegable (Fig. 8.32).

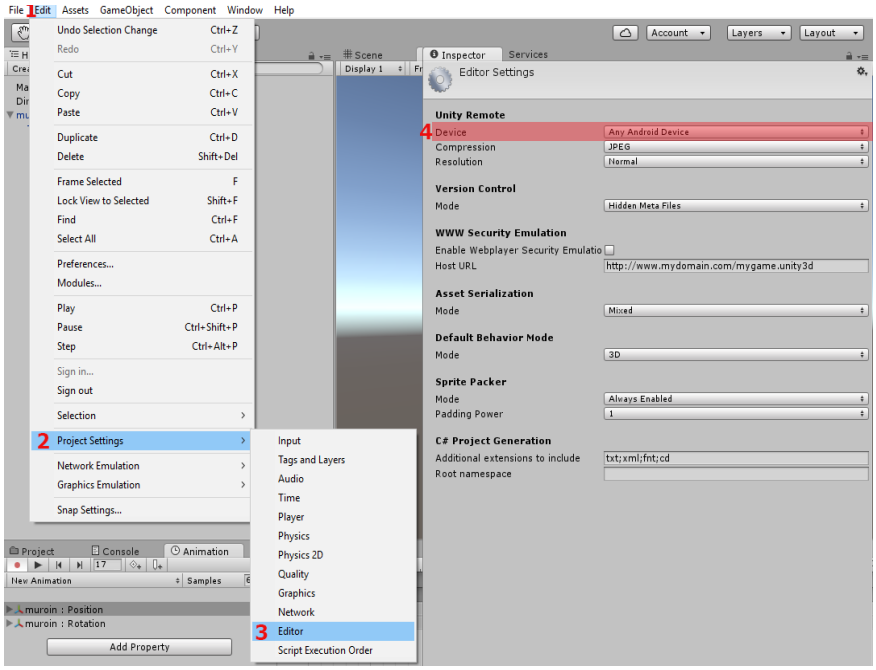


Figura 8.32. Editor settings Android.

Necesitamos definir el proyecto de Unity como Android, para ello nos dirigimos a Archivo, “Opciones de creación” o “Build Settings”. Pulsamos el boton de añadir las escenas abiertas como hemos realizado anteriormente. Después seleccionamos la plataforma de Android y pulsamos sobre el botón de “Cambiar plataforma”. Esperamos a que termine la barra de carga y pulsamos sobre el botón de “Ajustes del Reproductor” o “Player Settings” (Fig. 8.33).

En la parte derecha de la pantalla, en la pestaña de Inspector nos aparecen un buen número de opciones para configurar, tales como nombre de la aplicación, iconos y resolución a utilizar entre muchas otras (Fig. 8.34).

Los parámetros imprescindibles que necesitamos configurar para que no se genere ningun problema a la hora de compilar la aplicación y crear el archivo apk se encuentra en el apartado de “Otros Ajustes” (Fig. 8.34).

Necesitamos cambiar el Bundle Identifier que debe quedar de la siguiente forma com.Nombre_Compañia.Nombre_Aplicación (Fig. 8.34). Si dejamos este parámetro por defecto, no definimos un nombre de compañía y aplicación generara un error en el momento de compilar y no se creará el archivo apk.

En el apartado de Minimum API Level seleccionamos la versión de Android minima que se requerira para poder usar la aplicación, funcionará en versiones posteriores, pero no en anteriores, para este proyecto elegiremos la API 15 que corresponde a la version 4.0.3 Ice Cream Sandwich, debemos verificar que la versión de API que utilicemos previamente ha sido instalada con el SDK Manager, si no tendremos errores a la hora de compilar (Fig. 8.34).

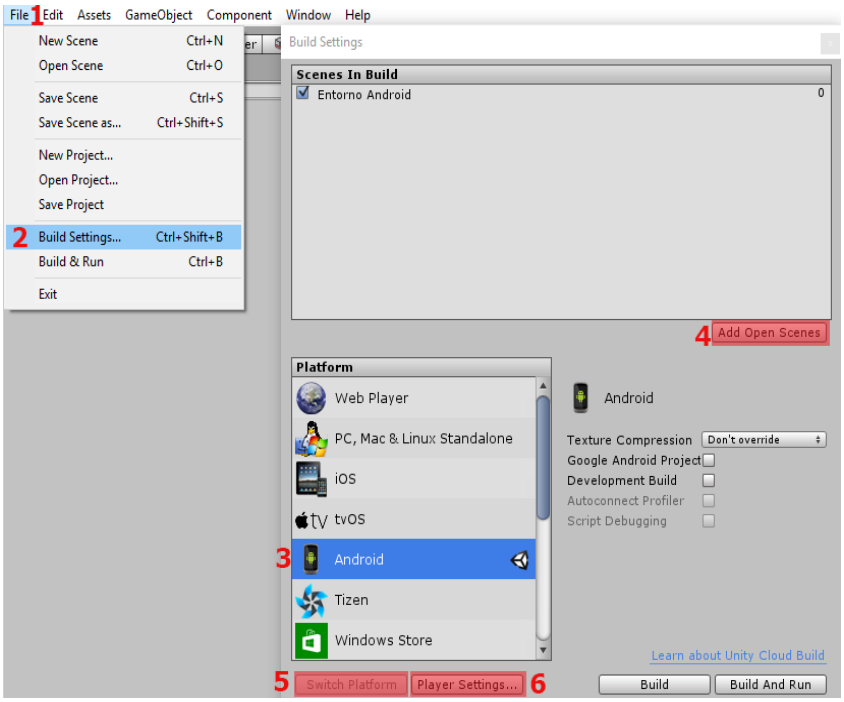


Figura 8.34. Build Settings Android.

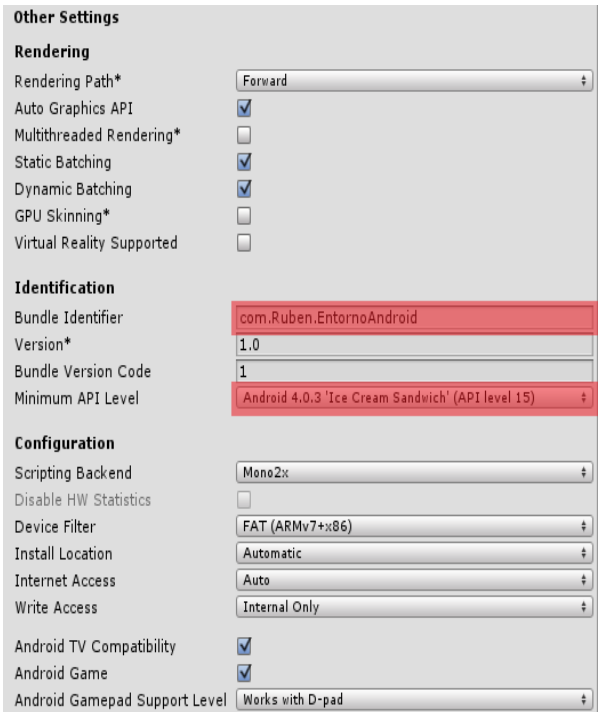


Figura 8.35. Other Settings Android.

Con ésto ya hemos configurado todo lo necesario para que nuestro proyecto pueda compilarse sin problemas para dispositivos Android. Simplemente quedaría generar el archivo .apk desde el menu de Build Settings que hemos visto anteriormente y presionar el botón “Build”. Seleccionar una carpeta donde queramos guardarlo, esperar a que finalice la barra de proceso e instalarlo en el dispositivo que queramos utilizarlo.

9. Conclusiones

En primer lugar tengo que expresar mi completa satisfacción personal por considerar como cumplidos todos y cada uno de los objetivos que me he propuesto en la elaboración del presente trabajo fin de grado.

Desde un primer momento he intentado y finalmente conseguido ser capaz de realizar de utilizar, importar y optimizar los datos extraidos de las nubes de puntos que podemos tomar mediante el láser escaner a motores gráficos.

Ademas de conseguir mediante la combinación de planos, escaner y fotografias obtener todos los datos necesarios para crear un entorno 3D lo más realista posible.

Mencionar que la principal problemática que presenta este proyecto es la elaboración en solitario del mismo. Recomiendo encarecidamente que para afrontar este tipo de trabajos se disponga de un equipo de trabajo coordinado y con una división clara de tareas. Ya que si se aborda en solitario puede resultar tedioso.

Gracias a la presente titulación, debemos ser pioneros en la creación de este tipo de productos. Disponemos de todos los conocimientos necesarios para generar grandes extensiones de entornos realistas, tales como: fotogrametría, teledeteccion, nubes de puntos 3D, toma de datos en campo y diseño de planos. Y tenemos la capacidad de ser precisos y rigurosos si el trabajo lo requiere.

Tambien considero que para generar entornos realistas se debe hacer una división del trabajo en: captura de datos, modelado, texturizado, programación y montaje del entorno. De esta forma se pueden crear entornos mucho mas complejos y detallados con la mayor calidad posible.

La toma de datos y diseño de planos nos ha permitido realizar el modelado del Castillo. La fotogrametría y nubes de puntos brindan la posibilidad de modelar grandisimas extensiones de una manera rapida y eficiente que no podriamos realizar de otra forma.

Conocimientos como la teledetección y fotogrametría nos ayudan en el correcto y preciso proceso de texturizado.

Los lenguajes de programación adquiridos han sembrado las bases del entendimiento del código y nos permiten abordar proyectos más complejos. Añadir funcionalidades para el usuario e incluso optimizar nuestro flujo de trabajo en ciertos aspectos.

Todos los conocimientos de gestión utilizados en SIG nos aportan la base para administrar grandes entornos con inmensidad de objetos. Teniendo la posibilidad de vincular la información, dotarla de posición y de esta manera gestionar el montaje todo el entorno.

Sin duda la elaboración del presente proyecto me deja completamente satisfecho, superando con creces mis espectativas a pesar de todas las dificultades encontradas.

Para terminar mencionar mi más profundo interes en seguir estudiando e investigando sobre cada uno de los aspectos que se han tratado aquí, intentar mejorar y crear nuevos proyectos de mayor calidad en el futuro.

10. Listado de figuras y tablas

11. Referencias

Group, M. (n.d.). *MeshLab User Guide*.

Valenzuela, E. A. (n.d.). Poisson Reconstruction.

Bolitho, M., Kazhdan, M., Burns, R., & Hoppe, H. (2009). Parallel poisson surface reconstruction. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and*

Gann, D. (1994). Archaeological site reconstruction with Autodesk's 3D Studio. CSA Newsletter (Center for the Study of Architecture), 7(3), 6–10.

Uray, F., Metin, A., & Varlik, A. (2015). 3D Architectural Surveying of Diyarbakir Wall's Ulu Beden Tower with Terrestrial Laser Scanner. Procedia Earth and Planetary Science, 15, 73–78. http://doi.org/10.1016/j.proeps.2015.08.019

Bétera, Ayuntamiento. (1998). Información del castillo de Bétera, 1–7.

Yuan, B., Goldman, R., Wang, E., Olorunnipa, O., & Khechoyan, D. (2016). Generating a 3D Normative Infant Cranial Model. Procedia Computer Science, 80, 988–998. http://doi.org/10.1016/j.pro

Engineering, I., Lin, K. H., Chang, C. H., Dopfer, a, & Wang, C. C. (2012). Mapping and Localization in 3D Environments Using a 2D Laser Scanner and a Stereo Camera. Journal of Information Science and Engineering, 28(2012), 131–144. Retrieved from <Go to ISI>://WOS:000299446100010

Guide, B. (2009). 3D Game Development with Microsoft Silverlight 3 Beginner ' s Guide. Packt Publishing. Retrieved from http://www.packtpub.com/3d-game-development-microsoft-silverlight-3-beginners-guide/book

Creighton, R. H. (2010). Unity 3D Game Development by Example. PACKT Publishing. Retrieved from https://www.packtpub.com/unity-3d-game-development-by-example-beginners-guide/book

Srivastava, S., & Chen, L. (2010). A two-parameter generalized Poisson model to improve the analysis of RNA-seq data. Nucleic Acids Research, 38(17), e170.

Smith, P., Hartley, T. P., & Mehdi, Q. H. (2013). C# interpreter and unity 3D for educational programming games. In Proceedings of CGAMES 2013 USA - 18th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational and Serious Games (pp. 41–47).

Moya-Maleno, P. R., Tormejón Valdelomar, J., Vacas Madrid, D., & Losa Sánchez, R. (2015). Interoperabilidad de la fotogrametría en modelado 3D: documentación, investigación y difusión en el yacimiento de Jamila. Virtual Archaeology Review, 6(13), 51–64.

ORTIZ SANZ, J., GIL DOCAMPO, M., MEIJIDE CAMESELLE, G., MARTÍNEZ RODRÍGUEZ, S., & REGO SANMARTÍN, M. T. (2011). Modelado 3D de una estación completa de petroglifos mediante escáner fotogramétrico de bajo coste: Pena de Chaos (Antas de Ulla, Lugo). Férvedes: Revista de Investigación, 7.

Ma, W., Hawkins, T., & Peers, P. (2007). Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. Proceedings of the 18th Eurographics Conference on Rendering Techniques, 183–194. http://doi.org/10.2312/EGWR/EGSR07/183-194

Osher, S., Tasdizen, T., Burchard, P., & Whitaker, R. (2003). Geometric surface processing via normal maps. ACM Transactions on Graphics, 22(4), 1012–1033.

Chen, A. (2012). Creating games with Unity and Maya: how to develop fun and marketable 3D games. Choice (Vol. 49). Retrieved from http://search.proquest.com/docview/1002728057?accountid=15533

Sun, F., System, J., & Server, A. (2008). The Java EE 5 Tutorial. *October*, *2010*(October), 821–1743. Retrieved from http://www.lavoisier.fr/notice/frEWO63O6AKRW32O.html