Ducke, Benjamin – Fritsch, Bernhard – Schilling, Manuel

Qualitätssicherung von 3D-Modellen für die Online-Publikation

aus / from

Forum for Digital Archaeology and Infrastructure, 1. Faszikel 2021, § 1-30

DOI: https://doi.org/10.34780/bi19-9w1a

Herausgebende Institution / Publisher:
Deutsches Archäologisches Institut

AN ARTICLE FROM THE

FDAI

FORUM FOR
DIGITAL ARCHAEOLOGY AND
INFRASTRUCTURE

ABSTRACT

Quality assurance of 3D models for online publication

Bernhard Fritsch, Manuel Schilling, in collaboration with Benjamin Ducke

Digital 3D models can be generated in many different ways and to different degrees of quality. But eventually, every 3D model should meet certain requirements to be classified publishable in an academic sense. These requirements pertain to visual quality as well as technical properties, including long-term archivability. In current practice, checking 3D models in regards to their quality before depositing and publishing in online repositories or databases is performed only rarely and manually. The use of free software allows to analyze 3D models automatically prior to publication, in order to meet minimal standards. In the following, an easy editing workflow that ensures the quality of a 3D model is presented.

Qualitätssicherung von 3D-Modellen für die Online-Publikation

unter Mitarbeit von Benjamin Ducke

Einleitung

1 Die stetig steigende Anzahl von 3D-Modellen, die im Rahmen einer Grabungsdokumentation oder auch für die Bearbeitung sonstiger archäologischer Fragestellungen hergestellt werden, fördert den Wunsch, diese Modelle auch online publizieren und wiederverwenden zu können[1]. Dementsprechend wächst die Anzahl an Repositorien und Online-Datenbanken, die fachspezifische 3D-Modelle vorhalten[2]. Allein aus Gründen der guten wissenschaftlichen Praxis sollte das Bestreben sowohl des Produzierenden eines 3D-Modells (im Weiteren: Datengebende:r) als auch des Betreibenden einer Onlineplattform (im Weiteren: Datennehmende:r) sein, die Modelle in bestmöglicher Qualität und mit so vielen dazugehörigen Informationen wie möglich (z. B. Texturen und ggf. Georeferenzierung) zur Verfügung zu stellen[3]. Die vielschichtigen Eigenschaften und Komponenten eines 3D-Modells können allerdings leicht zu, mitunter subtilen, Fehlern führen, die eine mangelhafte Darstellung des 3D-Modells nach sich ziehen[4]. Weiterhin können technische Ungenauigkeiten, die man auf den ersten Blick nicht erkennt, bei einer Nachnutzung zu großen Problemen führen.

1 Zu den weiteren Rahmenbedingungen, die eine Publikation von 3D-Daten erfüllen muss, um die Verfügbarkeit und Sicherheit der Daten im Sinne der Autoren zu gewährleisten, siehe Koller u. a. 2009.

2 Als Beispiele seien genannt: DARK Lab, Universität Lund (<https://models.darklab.lu.se/dynmcoll/Dynamic_Collections/>), Open Context (<https://opencontext.org/>), Global Digital Heritage (<https://globaldigitalheritage.org/>), iDAI.objects (<https://arachne.dainst.org/>), Edition Topoi (<http://repository.edition-topoi.org/>), Bonify (<https://www.digitalbones.eu/home>) (2.7.2021).

3 Für eine Übersicht über bisherige Projekte zu Repositorien mit 3D-Inhalten und deren unterschiedlichen Herangehensweisen an die praktische Umsetzung, Workflows und das Datenmanagement siehe Hardesty u. a. 2020.

4 Zu den technischen Schwierigkeiten, 3D-Daten überhaupt über das Internet anbieten zu können siehe Potenziani u. a. 2018.

2 Bezüglich einer problemlosen Darstellung von 3D-Modellen auf einer Website ist es oftmals anzuraten, für die Visualisierung eine gesonderte, komprimierte Version des 3D-Modells herzustellen. Dabei sollte die Originalversion ebenso gespeichert und zumindest zum Download verfügbar sein. Dieses Vorgehen und die verschiedenen Bauteile eines 3D-Modells, die als Einheit zusammenbleiben müssen, machen es notwendig, jedes 3D-Modell vor der Publikation einer Prüfung und Qualitätssicherung zu unterziehen. Nicht jede Redaktion (sofern überhaupt vorhanden) verfügt dabei über Fachkräfte, die den Umgang mit 3D-Modellen gewohnt sind und unter Umständen notwendige Bearbeitungen und Konvertierungen vornehmen können. Das Ziel ist es daher, einen möglichst automatisierten Prozess zu entwickeln, um die Aufnahme fehlerhafter Modelle in ein Online-Repositorium und unnötige Mehrarbeit zu vermeiden. Im Folgenden soll ein möglicher Weg aufgezeigt werden, der eine leichte und effiziente Qualitätssicherung und auch vorbereitende Bearbeitung für 3D-Modelle für eine Online-Publikation gestattet.

Idee

3 Im Fall von 3D-Modellen ist es möglich, ihre diversen Parameter und Eigenschaften computergestützt auszulesen. Sofern alle erforderlichen Vorgaben erfüllt sind und ein 3D-Modell somit als publikationswürdig eingestuft werden kann, ist es in der Regel notwendig, eine zweite Version des 3D-Modells für den jeweils verwendeten Webviewer anzulegen[5]. Unter anderem wird dafür das 3D-Modell komprimiert und in ein spezielles Format umgewandelt.

4 Beide Schritte können automatisch nacheinander mithilfe eines Skripts und durch die Nutzung weiterer Open-Source-Software durchgeführt werden. Die Ausführung des Skripts kann per Drag-and-Drop erfolgen, sodass keine weiteren Vorkenntnisse notwendig sind[6].

5 In dem hier betrachteten Beispiel wird dieser Prozess durch den Aufruf eines Skripts in der Programmiersprache Python ausgelöst und das Ergebnis der Untersuchung in einer Textdatei mit dem einfachen Format JSON (JavaScript Object Notation) ausgegeben. Die meisten Tests verwenden Funktionen der Python-Bibliothek PyMeshLab[7], während die nachfolgende Konvertierung und Komprimierung für Punktwolken unter Verwendung der Open-Source-Programme CloudCompare[8] und PotreeConverter[9] für den Online-Viewer Potree[10] erfolgen, sowie für 3D-Vermaschungen der Punktwolken (Meshes) mit dem Programm Nexus[11] für die Darstellung des Modells im Online-Viewer 3DHOP[12].

Vorgehensweise

6 Der Ablauf des gesamten Prozesses beginnt mit der Einreichung der 3D-Daten des:der Datengebenden (Abb. 1). Dabei sollten die 3D-Modelle bereits nach

5 Dieses Vorgehen entspricht auch den Prinzipien der »London Charter for the computer-based visualisation of cultural heritge« (bes. Principle 6 – Access), siehe auch Hermon – Niccolucci 2018, 44.

6 Voraussetzung ist, dass eine Python-Umgebung vorliegt. Das Skript »ObjectAnalyzer.py« liegt im Github-Bereich des DAI (< https://github.com/dainst/3D-integrity-check>) (30.7.2021).

7 PyMeshLab (<https://pymeshlab.readthedocs.io/en/latest/>) (2.7.2021).

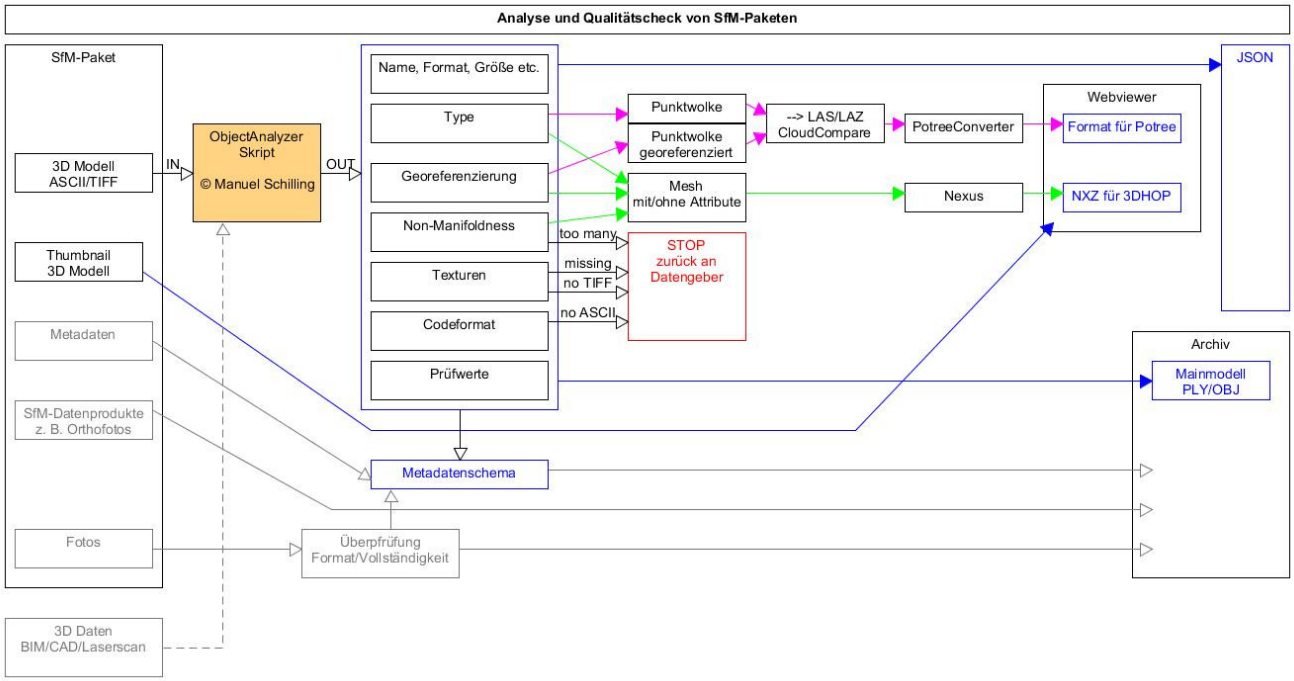
8 CloudCompare (<http://www.cloudcompare.org/>) (2.7.2021).

9 PotreeConverter (<https://github.com/potree/PotreeConverter>) (2.7.2021).

10 Potree (<https://github.com/potree/potree>) (2.7.2021).

11 Nexus (<http://vcg.isti.cnr.it/nexus/>) (2.7.2021).

12 3DHOP (<https://3dhop.net/>) (2.7.2021).



1

Abb. 1: Übersicht über den Ablauf des Qualitätschecks (Grafik: B. Fritsch)

Publikationsvorgaben des:der Datennehmenden aufbereitet sein. Das heißt u. a., dass das 3D-Modell mit ausreichend Metadaten[13] versehen ist und in einem passenden Dateiformat (idealerweise offen und für die Langzeitarchivierung geeignet) vorliegt.

7 Die korrekte Einhaltung der Vorgaben sowie die Vollständigkeit der Daten werden bei der Übergabe dann mithilfe des hier vorgestellten Skriptes überprüft. Das Ergebnis der Untersuchung wird in einer menschen- und maschinenlesbaren JSON-Datei ausgegeben. Sofern Probleme oder nicht erfüllte Vorgaben gefunden werden, bricht das Skript ab und das 3D-Modell selbst sowie die Gründe für die Ablehnung können in Form der JSON-Datei sozusagen als ›Quittung‹ an die oder den Datengebende:n zurückgeliefert werden.

8 Falls keine Probleme gefunden wurden, konvertiert das Skript das 3D-Modell anschließend in das passende, komprimierte Dateiformat für einen 3D-Online-Viewer, ohne die Originaldatei zu verändern. Letztere kann daher parallel in die Langzeitarchivierung überführt werden.

Eigenschaften des 3D-Modells

9 Im Einzelnen werden in der ersten Phase des Skriptes folgende Eigenschaften des 3D-Modells untersucht und dokumentiert: Neben dem Dateinamen, der Dateigröße und dem Dateityp wird angegeben, ob das Modell in einem Text- (ASCII-) oder binären Format gespeichert ist. Für eine Langzeitarchivierung der Daten ist eine Speicherung im ASCII-Format notwendig, sodass ein eingereichtes binäres Format bereits zu einer Ablehnung der Annahme des 3D-Modells führen würde.

13 Ebenso wie ein 3D-Modell aus mehreren Komponenten besteht, sind die notwendigen Metadaten für ein 3D-Modell aufgrund des Entstehungsprozesses mehrschichtig und ausführlich. Korrekte Metadaten sind dennoch ein wichtiger Bestandteil, um die Qualität eines 3D-Modells beurteilen zu können. Zu der Frage eines diesbezüglichen Metadatenschemas siehe Homburg u. a. 2021.

a b c

2

Abb. 2: Vermaschtes, georeferenziertes 3D-Modell eines Grabungsschnittes (a). Dasselbe Modell mit fehlender Texturdatei (b) und mit fehlerhafter Darstellung der Georeferenzierung in einem Webviewer (c) (Grafik: B. Fritsch)

10 Der zweite Block listet die Anzahl der *Vertices* (Punkte) und *Faces* (Flächen) des 3D-Modells auf, woraus wiederum auf die Art des 3D-Modells geschlossen werden kann. Bei einer Punktwolke ist die Anzahl der *Faces* gleich Null. Diese Information ist hilfreich, um das 3D-Modell einem geeigneten 3D-Viewer zuführen zu können beziehungsweise die für den jeweiligen Viewer spezifische Komprimierung des Modells einzuleiten.

11 Ebenso wird eine mögliche Georeferenzierung anhand der Länge der Punktkoordinaten untersucht. Auch das ist eine entscheidende Information hinsichtlich der Auswahl und Aufbereitung des 3D-Modells für den Online-Viewer. Georeferenzierte Modelle können aufgrund der verwendeten OpenGL- oder WebGL-Technologie der Online-Viewer derzeit nicht korrekt dargestellt werden[14] und müssen dementsprechend vorher in ein passendes Format umgewandelt werden, ohne die wertvollen Geoinformationen zu verlieren (Abb. 2c). Ferner sieht das Skript auch eine Berechnung der Qualitätsmetriken *non-manifold-edges* und *non-manifold-vertices* vor. Diese zeigen die tatsächliche topologische Qualität eines 3D-Modells auf. Je mehr *non-manifold*-Elemente, wie beispielsweise sich schneidende *Faces* oder zu viele an derselben Kante anliegende Flächen, vorliegen, desto schlechter ist die Geometrie des Modells. Auch wenn das vom menschlichen Auge in der Regel nicht sofort erkannt wird[15], kann es zu großen Problemen bei der Komprimierung des Modells für eine flüssige Webansicht führen oder auch einen 3D-Druck unmöglich machen. Diese Problematik taucht bei punktwolkenbasierten 3D-Modellen nicht so häufig auf wie bei NURBS-basierten Modellen[16], da die Scan-Verfahren theoretisch nur Oberflächen abbilden können. Trotzdem ist eine Kontrolle der Geometrie von 3D-Modellen jeglicher Art vor einer Publikation anzuraten[17].

12 Sehr häufig allerdings stellen Texturen, die in extra Bilddateien gespeichert werden, eine mögliche Fehlerquelle dar. Dies liegt schlicht daran, dass Texturen durch eine Umbenennung oder ein Vergessen bei einem Kopiervorgang nicht mehr mit dem 3D-Modell verknüpft sind und dadurch das Modell visuell falsch dargestellt wird

14 Eine kurze Erläuterung zu diesem Themenkomplex findet sich in der Beschreibung des von CloudCompare verwendeten Workarounds: Global Shift and Scale (<https://www.cloudcompare.org/doc/wiki/index.php?title=Global_Shift_and_Scale>) (21.7.2021).

15 Solche problematischen Stellen finden sich häufig im ›Inneren‹ eines Modells, sodass der äußere Eindruck des Modells korrekt erscheint.

16 Dies sind Modelle, die bspw. mit Programmen wie AutoCAD, Cinema4D, Blender o. ä. erstellt werden.

17 Das Skript stoppt momentan bereits beim Auffinden eines *non-manifold*-Elementes. Hier wäre das Einsetzen eines Toleranzwertes denkbar.

(Abb. 2b). Da ein Modell unter Umständen mit mehreren Texturdateien – und im Fall des bei Archäolog:innen gerne verwendeten OBJ-Formats noch zusätzlich mit einer obligatorischen Materialdatei – verknüpft sein kann, erzeugt eine solche Konstellation aus vielen getrennten Dateien ein erhöhtes Risiko für Fehler. Die Vollständigkeit des gesamten Paketes muss sowohl für eine korrekte Publikation und Visualisierung (Abb. 2a) als auch für eine Weitergabe und Nachnutzung der Modelle unbedingt gegeben sein.

13 Die Anzahl und die Namen der Texturdatei(en) sind in der eigentlichen 3D-Datei vermerkt und können daher von dem Skript ausgelesen werden. Darauf folgt eine Überprüfung, ob die erwähnten Dateien auch im selben Ordner, in dem die Überprüfung stattfindet, vorhanden sind. Weiterhin kann überprüft werden, ob die Texturdateien im TIFF-Format vorliegen. Dieses Format ist für die Darstellung des Modells und weitere Bearbeitungen zwar nicht relevant, aber als notwendig anzusehen, falls das Modell einer Langzeitarchivierung zugeführt werden soll. Denn zu diesem Zweck ist das TIFF-Format allgemein das für Bilddateien empfohlene Format[18].

14 Schließlich ist es noch möglich, eindeutige digitale ›Fingerabdrücke‹ (Hashwerte) oder eine Prüfsumme (Checksum) für das 3D-Modell beziehungsweise das aus Modell und Texturdateien bestehende 3D-Paket zu berechnen und auszugeben. Diese Werte sind nützlich, um Änderungen – und damit Fehlerquellen – schnell am 3D-Modell festzustellen[19]. Solche Änderungen können u. a. durch Kopierprozesse entstehen, beispielsweise, wenn ein Repositorium zu einem neuen, größeren Speicherort überführt wird oder die Dateien aus anderen Gründen innerhalb eines Systems verschoben werden müssen. Momentan benutzt das Skript die Open-Source-Programmbibliothek Hashlib und gibt einen MD5-Wert aus. Allerdings unterliegen diese Analysen einer ständigen Weiterentwicklung und Anpassung, sodass dieser Punkt bei einer konstanten Nutzung des Qualitätssicherungs-Skriptes beobachtet und gegebenenfalls angepasst werden muss.

Webviewer

15 Sobald die Überprüfung des 3D-Modells für die genannten Parameter abgeschlossen ist, steht fest, ob das Modell in der vorliegenden Form publiziert werden kann, oder ob es noch Nacharbeiten und Korrekturen seitens des:der Datengeber:in bedarf.

16 Sofern keine Änderungen mehr vorgenommen werden müssen, wird automatisch der nächste Schritt ausgeführt. Dies bedeutet, dass von dem Modell eine zweite Version in einem bestimmten Dateiformat angefertigt wird, das für einen Webviewer optimiert ist. Dafür werden die Modelle in ihrer Dateigröße komprimiert und in ein spezielles Dateiformat konvertiert, ohne relevante Informationen zu verlieren. Das vorliegende Skript kann diesen Prozess zum einen für den Webviewer Potree, der speziell für große Punktwolken geeignet ist, und zum anderen für den Webviewer 3DHOP, der kleinere Punktwolken und Meshes mit oder ohne Texturdateien optimal darstellt, durchführen.

17 Beide Viewer sind als Open Source verfügbar und enthalten mehrere Tools, wie zum Beispiel ein Messwerkzeug oder die Möglichkeiten, Querschnitte durch das

18 Siehe die IT-Empfehlungen bei Ianus (<https://www.ianus-fdz.de/it-empfehlungen/dateiformate>) (21.7.2021) oder dem Archaeological Data Service (<https://archaeologydataservice.ac.uk/advice/Downloads.xhtml>) (21.7.2021).

19 Siehe Nestor 2010, Kapitel 9.3.

Modell zu legen. Da der Viewer Potree, beziehungsweise der PotreeConverter, der die Konvertierung in das Potree-Format vornimmt, ausschließlich mit LAS/LAZ-Dateien funktioniert, werden die dafür geeigneten Punktwolken vorher mit der Open-Source-Software CloudCompare in das passende Format umgewandelt. Das funktioniert sowohl mit PLY- als auch OBJ-Dateien, die die beiden gängigsten Formate im archäologischen Bereich sind.

18 Wird in dem Skript das Vorliegen eines Meshes (z. B. einer aus den 3D-Punkten per Dreiecks-Vermaschung rekonstruierten Oberfläche) festgestellt, erfolgt die Nutzung des Programmes Nexus[20]. Nexus ist Teil des 3DHOP-Angebotes und wandelt Meshes, wiederum aus dem PLY- oder OBJ-Format, mit oder ohne Textur in das Format NXS/NXZ um[21]. Die resultierenden Dateien sind deutlich kleiner als das Originalmodell und fassen alle Informationen in einer Datei zusammen. Der Webviewer 3DHOP schließlich zeigt diese Dateien sehr schnell und problemlos in jedem gängigen Browser an.

19 Beide Methoden bieten durch die Konvertierung die Möglichkeit, auch georeferenzierte Modelle im Webviewer korrekt abzubilden. Da die vollständigen Geoinformationen dann aber nicht mehr direkt aus der Präsentationsversion ausgelesen werden können, ist es notwendig, die Original- oder Archivversion des 3D-Modells zum Download anzubieten. Mit Hilfe des Skriptes ist der Aufwand dafür äußerst gering, weil die notwendigen Schritte hier eingebunden sind und die Präsentationsversionen als Kopie des Originals hergestellt werden. Nach Ausführen des Skriptes liegen also sofort beide Versionen vor, die für eine Archivierung und Online-Publikation notwendig sind.

Ergebnis

20 Ausgangspunkt für einen Qualitätscheck von 3D-Modellen sind das hier publizierte Python-Skript und alle notwendigen Open-Source-Komponenten (PyMeshLab, Nexus, CloudCompare, PotreeConverter) sowie das 3D-Modell, das untersucht werden soll, ebenfalls mit allen notwendigen Komponenten, wie zum Beispiel Texturdateien. Nach der Ausführung des Skriptes sind zwei Resultate möglich:

1. Das Skript findet Fehler oder fehlende Informationen. In diesem Fall wird nur eine JSON-Datei ausgegeben (Abb. 3) und keine weitere Bearbeitung des Modells hinsichtlich eines Webviewers vorgenommen. In der JSON-Datei lässt sich die Angabe zum Grund des Abbruchs leicht finden: Entweder die Datei liegt nicht im ASCII-Format vor, es fehlen Texturdateien oder es existieren *non-manifold*-Elemente. Das 3D-Modell sollte folglich an die:den Datengebende:n zurückgegeben und korrigiert werden.
2. Das Skript findet keine Fehler und startet auf Basis der vorliegenden Informationen die Konvertierung in ein geeignetes Format für einen Webviewer (Potree oder Nexus), unter Beachtung der verwendeten Texturen und gegebenenfalls der Georeferenzierung. In diesem Fall liegt nach Beendigung der Skript-Ausführung neben der JSON-Datei mit allen Informationen (Abb. 4) zusätzlich noch die komprimierte Version des 3D-Modells am selben Speicherort vor.

20 Eine Alternative zu Nexus bietet die Open-Source-Programmbibliothek Draco (<https://google.github.io/draco/>) (21.7.2021).

21 Seit der neuesten Version von MeshLab (Released: 23.7.2021) ist ein Nexus-Export der Modelle direkt aus MeshLab möglich und muss nicht in einer eigenen Software ausgeführt werden. Dieser Schritt ist dadurch sehr leicht vom Datengebenden durchzuführen. Nur im Fall von georeferenzierten Modellen muss weiterhin die klassische Nexus-Version benutzt werden, weil diese Modelle von MeshLab nicht korrekt geladen und folglich nicht konvertiert werden können.

```
{
    "originally": {
        "Objectname": "Ostia201Mesh",
        "Objectformat": ".obj",
        "Codeformat": "ascii",
        "Objectsize": 102301775,
        "hash code": "aabf9eb5ac7d360fb043acb571f34ccb"
    },
    "characteristics": {
        "Type": "mesh",
        "Vertices": 665105,
        "Faces": 1327522,
        "Edges": 0,
        "Georeferences": false
    },
    "non manifoldness": {
        "non manifold edges": false,
        "non manifold vertices": 2
    },
    "materials": {
        "Name/Path": "Ostia201Mesh.mtl",
        "Found": true,
        "Size": 177,
        "textures": {
            "Number of required textures": 1,
            "Number of found textures": 0,
            "Number of missing textures": 1,
            "Number of TIF textures": 0,
            "Number of missing TIF textures": 1,
            "texture_1": {
                "Name/Path from MTL": "Ostia201Mesh.jpg
                "Found": false
            }
        }
    }
}
```


3


```
{
    "originally": {
        "Objectname": "Ostia201_3D",
        "Objectformat": ".obj",
        "Codeformat": "ascii",
        "Objectsize": 122835493,
        "hash code": "c2202f26cd2edecc167400646971cbc4"
    },
    "characteristics": {
        "Type": "mesh",
        "Vertices": 665105,
        "Faces": 1327522,
        "Edges": 0,
        "Georeferences": true
    },
    "non manifoldness": {
        "non manifold edges": false,
        "non manifold vertices": 2
    },
    "materials": {
        "Name/Path": "Ostia201_3D.mtl",
        "Found": true,
        "Size": 177,
        "textures": {
            "Number of required textures": 1,
            "Number of found textures": 1,
            "Number of missing textures": 0,
            "Number of TIF textures": 1,
            "Number of missing TIF textures": 0,
            "texture_1": {
                "Name/Path from MTL": "Ostia201Mesh.tif",
                "Found": true,
                "Size": 50331906
            }
        }
    }
}
```


4

21 Die Arbeitsschritte sowie die richtige Beurteilung der Ergebnisse in Form der JSON-Datei oder der durchgeführten Erstellung des Präsentationsmodells können ohne Vorkenntnisse im Bereich 3D durchgeführt werden. Es ist zu empfehlen, die Informationen aus der JSON-Datei im Anschluss in die Metadaten zu übertragen (sofern die Angaben wie Dateiname und -größe nicht schon vorliegen), oder die Datei als Ganzes zusammen mit dem 3D-Modell am selben Ort zu speichern. Hierdurch können auch spätere Kontrollen oder Abfragen erfolgen, ohne das 3D-Modell mit einem extra Programm öffnen zu müssen.

Abb. 3: Beispiel der Ausgabe der Ergebnisse in einer JSON-Datei. Die fehlende Texturdatei führt zu einem Abbruch des Prozesses (Grafik: B. Fritsch)

Abb. 4: Beispiel der Ausgabe der Ergebnisse in einer JSON-Datei. Das 3D-Modell erfüllt alle Kriterien und die Konvertierung in ein Präsentationsmodell wurde vorgenommen (Grafik: B. Fritsch)

Ausblick

22 Besonders bei Plattformen, die 3D-Daten aus unterschiedlichen Disziplinen und Forschungsbereichen anbieten, kommt es häufig vor, dass sich die Modelle selbst in grundlegenden Eigenschaften stark unterscheiden. Die Modelle können mit unterschiedlichen Methoden erstellt worden sein oder, je nach Situation oder Objekt, verschiedene Schwerpunkte haben. Manchmal kann eine Georeferenzierung wichtiger sein als eine klare und scharfe Textur und umgekehrt. Dementsprechend können die Dateien auch in diversen Dateiformaten vorliegen und auch der Grad ihrer Bearbeitung kann unterschiedlich sein. In manchen Fällen ist eine Punktwolke ausreichend oder sogar besser für computergestützte Analysen des 3D-Modells[22]; manchmal kann nur

22 Siehe z. B. Nobles – Roosevelt 2021. Die Studie sieht die Publikation und Visualisierung von 3D-Modellen nicht als letztes zu erreichendes Ziel an, sondern verwendet die Punktwolken für weitere Untersuchungen auch hinsichtlich eines 3D-GIS.

anhand eines Meshes und einer detaillierten, farbechten Darstellung der Oberfläche die archäologische Fragestellung angegangen werden[23].

23 Trotzdem müssen einige grundlegende Voraussetzungen für eine Publikation eines 3D-Modells erfüllt sein, da ein 3D-Modell sonst nicht weiterverwendet werden kann beziehungsweise eine transparente Nachvollziehbarkeit der Daten im Sinne der FAIR-Prinzipien[24] nicht gegeben ist.

24 Abgesehen von der Menge der Daten und dem Aufwand, eine große Anzahl von 3D-Modellen für eine Publikation bzw. Archivierung in einem Online-Repositorium vorzubereiten, ist es letztlich auch für die Nutzer:innen der Daten viel angenehmer und ansprechender, wenn die publizierten Daten gewisse visuelle und qualitative Ansprüche erfüllen und ein 3D-Modell etwa nicht völlig falsch orientiert im Raum schwebt. Ein gewisser Teil dieser Voraussetzungen kann mit dem vorgestellten Skript sichergestellt werden, ohne einem:einer Bearbeiter:in viel zusätzliche Arbeit zu schaffen.

25 Es ist aber festzuhalten, dass die Möglichkeiten einer derartigen Qualitätssicherung bei weitem noch nicht ausgeschöpft sind und sich möglicherweise hilfreiche Python-Libraries ständig weiterentwickeln oder auch neue hinzukommen. Ein solches Tool oder andere ähnliche Skripte sollten also immer wieder aktualisiert und verbessert werden[25]. Sofern solche Tools aber in einen redaktionellen Prozess aufgenommen werden, stellen sie auch zum jetzigen Zeitpunkt eine große Hilfe dar, um die wissenschaftliche Publikation von 3D-Modellen voranzubringen.

26 Komplett in der Verantwortung des:der Datengebenden bleibt weiterhin eine korrekte Skalierung des 3D-Modells – eine weitere Grundvoraussetzung eines publikationsfähigen 3D-Modells. Der Ursprung oder die Art des in 3D abgebildeten Originals kann von einem Computer (noch) nicht selbstständig bestimmt werden.

Benutzung

27 Das hier vorgestellte Skript wurde in der Programmiersprache Python 3 geschrieben. Die für die technischen Abfragen des 3D-Modells genutzte Library PyMeshLab existiert erst seit Kurzem und wird noch weiterentwickelt. Für die Nutzung des Skriptes müssen Python sowie die Programme Nexus, PotreeConverter und CloudCompare installiert sein.

28 Das Skript und die verwendeten Programme können als Gesamtpaket in einem Ordner unabhängig von möglicherweise anderen installierten Versionen auf einem Computer abgelegt und dort ausgeführt werden. Die zu untersuchenden 3D-Modelle müssen dann in diesen Ordner kopiert werden. Das Modell kann entweder per Drag-and-Drop auf das Skript gezogen werden, oder das Skript wird in einer gewohnten Python-Umgebung über einen Run-Befehl ausgeführt.

23 Dabei können die unterschiedlichen Herangehensweisen selbst wiederum verschiedene Aspekte aufzeigen, siehe dazu Atteni u. a. 2017 zu einem Vergleich von SfM-Texturen und RGB-Daten von Laserscans.

24 Dies betrifft im Besonderen den Punkt R1.2 der FAIR-Prinzipien (<https://www.go-fair.org/fair-principles/r1-2-metadata-associated-detailed-provenance/>) (21.7.2021).

25 Eine sehr hilfreiche Ergänzung wäre eine automatisierte Prüfung von 2D-Bilddateien. Diese könnte für Bilder an sich, aber auch für Bilder aus einem SfM-Workflow verwendet werden und so die Qualitätssicherung von SfM-3D-Modellen vervollständigen.

29 Folgende Abfragen werden ausgeführt und in einer JSON-Datei als eine Art Quittung ausgegeben (Abb. 5). Die vorliegende Speicherung im binären Format, das Fehlen von Material- beziehungsweise Texturdateien und das Vorhandensein von *non-manifold*-Elementen erzwingt einen Abbruch des Skriptes.

30 Ein den Vorgaben entsprechendes 3D-Modell wird in das passende Dateiformat für den jeweiligen 3D-Webviewer konvertiert und in entsprechenden (neuen) Ordnern abgelegt.

Kategorie	Information
Dateiname	
Dateiformate	
Codeformat	Binär oder ASCII
Dateigröße	
Typ	Mesh oder Punktwolke
Anzahl Vertices	
Anzahl Faces	
Georeferenzierung	True/False
Non-Manifoldness	Anzahl von non-manifold-edges und non-manifold-vertices
Materialdatei	True/False
Texturdateien	Anzahl, Dateiformat, Anzahl fehlender Dateien

5

Abb. 5: Übersicht der durch das Skript erfolgten Abfragen (Tabelle: B. Fritsch)

Referenzen

ObjectAnalyzer 202 Github-Repository: 3D-integrity-check

Atteni u. a. 2017 M. Atteni – V. Caniglia – C. Inglese – A. Ippolito, Quality vs. Quantity. Advantages and Disadvantages of Image-Based Modeling, in: J. B. Glover – J. Moss – D. Rissolo (Hrsg.), CAA – Digital Archaeologies, Material Worlds (Past and Present). Proceedings of the 45rd Annual Conference on Computer Applications and Quantitative Methods in Archaeology (Tübingen 2020)

Hardesty u. a. 2020 J. Hardesty – J. Johnson – J. Wittenberg – N. Hall – M. Cook – Z. Lischer-Katz – Z. Xie – R. McDonald, 3D Data Repository Features, Best Practices, and Implications for Preservation Models. Findings from a National Forum. College & Research Libraries 81(5), 2020, 789

Hermon – Niccolucci 2018 S. Hermon – F. Niccolucci, Digital Authenticity and the London Charter, in: P. Franco – F. Galeazzi – V. Vassallo (Hrsg.), Authenticity and cultural heritage in the age of 3D digital reproductions, 2018, 37–47

Homburg u. a. 2021 T. Homburg – A. Cramer – L. Raddatz – H. Mara, Metadata schema and ontology for capturing and processing of 3D cultural heritage objects, Herit Sci 9, 2021, 91

Koller u. a. 2009 D. Koller – B. Frischer – G. Humphreys, Research challenges for digital archives of 3D cultural heritage models, in: Journal on Computing and Cultural Heritage, Volume 2, Issue 3, 2009, 1–17

Nestor 2010 H. Neuroth – A. Oßwald – R. Scheffel – S. Strathmann – K. Huth (Hrsg.), nestor-Handbuch. Eine kleine Enzyklopädie der digitalen Langzeitarchivierung, Version ²(Göttingen 2010)

Nobles – Roosevelt 2021 G. R. Nobles – C. H. Roosevelt, Filling the Void in Archaeological Excavations. 2D Point Clouds to 3D Volumes, Open Archaeology 7.1, 2021, 589–614

Potenziani u. a. 2018 M. Potenziani – M. Callieri – M. Dellepiane – R. Scopigno, Publishing and Consuming 3D Content on the Web. A Survey, Foundations and Trends® in Computer Graphics and Vision 10.4, 2018, 244–333

AUTHORS

Dr. Bernhard Fritsch
Deutsches Archäologisches Institut, Zentrale Wissenschaftliche Dienste (ZWD)
Podbielskiallee 69–71
14195 Berlin
Deutschland
bernhard.fritsch@dainst.de
ORCID-iD: https://orcid.org/0000-0003-1858-5070
ROR: https://ror.org/041qv0h25

Manuel Schilling
HTW Dresden
Friedrich-List-Platz 1
01069 Dresden
Deutschland
man.r.schilling@gmail.com
ROR: https://ror.org/05q5pk319

Dr. Benjamin Ducke
Deutsches Archäologisches Institut, Zentrale Wissenschaftliche Dienste (ZWD)
Podbielskiallee 69–71
14195 Berlin
Deutschland
benjamin.ducke@dainst.de
ORCID-iD: https://orcid.org/0000-0002-0560-4749
ROR: https://ror.org/041qv0h25

METADATA

Titel/*Title*: Qualitätssicherung von 3D-Modellen für die Online-Publikation
Band/*Issue*: FdAI 2021/1

Cover Illustration: Grafik: Bernhard Fritsch, DAI, Zentrale Wissenschaftliche Dienste

Bitte zitieren Sie diesen Beitrag folgenderweise/ *Please cite the article as follows*:
B. Fritsch – M. Schilling – B. Ducke, Qualitätssicherung von 3D-Modellen für die Online-Publikation, FdAI 2021/1, § 1–30, https:/doi.org/10.34780/bi19-9w1a

Online veröffentlicht am/*Online published on*: 15.12.2021
DOI: https://doi.org/10.34780/bi19-9w1a
URN: https://nbn-resolving.org/urn:nbn:de:0048-bi19-9w1a.9

Bibliographic reference: https://zenon.dainst.org/Record/002059992

JOURNAL METADATA

Forum for Digital Archaeology and Infrastructure published since 2021

Publisher/Editors
Benjamin Ducke, Friederike Fless, Reinhard Förtsch, Fabian Riebschläger, Henriette Senst
Deutsches Archäologisches Institut
Podbielskiallee 69–71
14195 Berlin
Deutschland
http://www.dainst.org

Editing and Typesetting
Publishing editor: Deutsches Archäologisches Institut, Zentrale – Stabsstelle Kommunikation, Redaktion
Editing: Antonie Brenne, Janina Rücker M.A. (fdai-journal@dainst.de)

Corporate Design: LMK Büro für Kommunikationsdesign, Berlin
Webdesign: LMK Büro für Kommunikationsdesign, Berlin (lm-kommunikation.de)
Programming Viewer: LEAN BAKERY, München (leanbakery.com)

Cover illustration: R. Mecking – E. Erkul – W. Rabbel (CAU Kiel, Institut für Geophysik); Topographische Daten: Geographisches Institut der Universität zu Köln (A. Bolten – H. Brückner) – DAI-Pergamongrabung; Gestaltung: LMK Büro für Kommunikationsdesign, Berlin