

Guide for teachers / trainers / guides using Mixed Reality outdoor educational tool

28 February 2019



Paper manual and tool guide (tutorial) were elaborated by:

Technical University of Madrid

KGHM Cuprum Ltd. Research & Development Centre

Dnipro University of Technology

KAVA Reference (Number, Acronym, Full Title): **16373, VirtualMine, VirtualMine - as a modeling tool for Wider Society Learning**

Name of the author/Responsible partner: **Francisco Pedro Luque Oostrom, CeDInt - Technical University of Madrid**

Version No: **1**

Content

1.	General description.....	3
2.	Installation.....	3
3.	Server configuration.....	4
4.	Content generation	4
4.1.	SQLite database.....	5
4.2.	3D models	7
4.3.	Additional media content.....	10
4.4.	Map textures.....	10
4.5.	Vuforia AR markers	12
4.6.	AssetBundle generation	16
5.	User interface.....	17
6.	Position Simulator	23
7.	Examples of 3D content – selected models	25

1. General description

This application “**Mixed Reality outdoor educational tool**” consists of an AR / MR educational tool allowing real-time interaction between a group of children/students/adults and the teacher/guide. The classes would take place in an open field in the area of former mining excavations (underground or surface). The beginning of the workshops would start by showing a map marker to the device camera. As a result, an interactive map will be shown with points of interest (POIs) marking places of processing, storage of ore, tailings collection, etc. Each of them would be possible to click to start the navigation to that location. In addition, the teacher would have the opportunity to draw a navigation path or mark other interesting places on this map. After discussing the region of former exploitation, a field game would be started, consisting in finding further remains of the former mining.

Technical assumptions and contents of substantive content including 3D models and processes visualizations (animations) for 2 pilot areas (Marcelo Jorissen Mine in Madrid and Tourist Route “By the traces of the former ore mining” in the Mirsk Commune, Poland) were developed.

2. Installation

In the *VirtualMine_FastTrack.zip* file you should find the following elements:

- **VirtualMineFastTrack.apk**: this file should be copied and installed to an android mobile device capable of running the application (it requires that this mobile device is provided with a high resolution camera, GPS and compass sensors, and a medium-high graphic processing unit). Permission to access camera and user location must be granted while installing the application.
- **\FastTrackBundleGenerator**: this folder is a Unity3D project (version 2017.4.1.f1) used to edit and generate compatible content for the application. A description of the process to create and upload content to the server is described in detail in section 4. An scene called “BundleGenerator_MarceloJorissen” has been included as an example on how to generate content.
- **\Markers**: printable AR markers for each former mine location.
- **VirtualMine_FastTrack_Manual_vXX.docx**: this manual document.

3. Server configuration

A HTTP server site is required for the application to work and download content dynamically. A virtual host with the name “virtualmine” has to be enabled in this server and all generated contents (using the provided *FastTrackBundleGenerator* Unity project) must be uploaded to this virtual host from the following folder inside the project:

..\FastTrackBundleGenerator\Assets\virtualmine\

Currently, CeDint-UPM has a working server in the following URL:

<http://viz2know.cedint.upm.es/virtualmine/>

Important: at runtime, only the server name (e.g. viz2know.cedint.upm.es) has to be filled in the corresponding input field of the server configuration panel (see section 5).

4. Content generation

To generate content for the application, use the *FastTrackBundleGenerator* project with Unity3D Editor (version 2017.4.1.f1). Inside the Assets folder, only the following elements should be updated or edited to generate new content:

- **/Databases/VirtualMine.bytes:** this is the SQLite database file with all the configuration, locations and descriptions for the application. This file should be edited using an external SQLite browser application (e.g. DB Browser for SQLite). When updated, overwrite this file first before generating the database AssetBundle as described in section 4.6. (See also section 4.1 for a complete database description)
- **/Models/[LocationName]/[BundleName]/:** in each [LocationName] in the *VMLocation* table of the SQLite database, there is a folder for each 3D model referred as [BundleName] in the *Bundle* table of the SQLite database. The procedure to create 3D models is described in section 4.2.
- **/Media/[BundleName]/:** this folder contains the additional Media content for the AR markers of each POI. The procedure to create additional media content is described in section 4.3.
- **/Textures/:** this is a collection of map textures used as maps over the AR map markers within the application. Each texture is identified with a name that must be the same as configured in the SQLite database (“Texture” field in the “VMLocation” table). The procedure to create map textures is described in section 4.4.
- **/Markers/:** this folder contains the Vuforia AR marker database. For each [LocationName], there is a Vuforia database built with two files ([LocationName].dat and

[LocationName].xml). The procedure to generate these database files is described in section 4.5.

4.1. SQLite database

The SQLite database is formed by 4 different tables to configure the application. Use a SQLite browser application (e.g. SQLite DB Browser for SQLite) to edit this file easily. The fields for each table is described below.

- **VMLocation:** this table contains the former mine location areas configuration for the application.

ID (Integer)	The ID of this mine location to identify in other tables
Name (string)	The name code of this mine location. This code should match the value of [LocationName] in the /Models/[LocationName] folders and /Markers/[LocationName].[dat/xml] files
LongName	The name string to be shown in the User Interface of the application
BL_Latitude (double)	The latitude in decimal format of the bottom left corner of the map texture for this mine location.
BL_Longitude (double)	The longitude in decimal format of the bottom left corner of the map texture for this mine location.
TR_Latitude (double)	The latitude in decimal format of the top right corner of the map texture for this mine location.
TR_Longitude (double)	The longitude in decimal format of the top right corner of the map texture for this mine location.
Texture (string)	The name of the texture to use as map in the /Textures/ folder. If <i>NULL</i> then no texture will be used and the AR map marker will be used as map texture. This is useful if the marker has enough quality to be detected by Vuforia Engine (see Vuforia database subsection).

- **Marker:** this table contains the POIs configuration for each mine location

ID (Integer)	The ID of this POI to identify uniquely in other tables.
Name (string)	The name that should appear next to the location marker in the application.
Latitude (double)	The latitude in decimal format for this POI.
Longitude (double)	The longitude in decimal format for this POI.
LocationID (int)	The ID of the mine location area referred in the <i>VMLocation</i> table of the database.
BundleID (int)	The ID of the 3D model referred in the <i>Bundle</i> table of the database. If this value is zero, then no 3D model will be loaded for this POI.
Scale (double)	The scaling value to use for the 3D model when the AR marker is pointed with the camera.
Position (string)	The Unity local coordinates for the 3D model when the AR marker is pointed with the camera. The format is: [x coord];[y coord];[z coord]
Rotation (string)	The Unity local rotation in Euler angles for the 3D model when the AR marker is pointed with the camera. The format is [x rot];[y rot];[z rot]
DescriptionID	The ID of the POI description referred in the <i>MarkerDescription</i> table of the database.
MediaBundleID	The ID of the media content referred in the <i>Bundle</i> table of the database.

- **Bundle:** this table contains the 3D models and additional media names

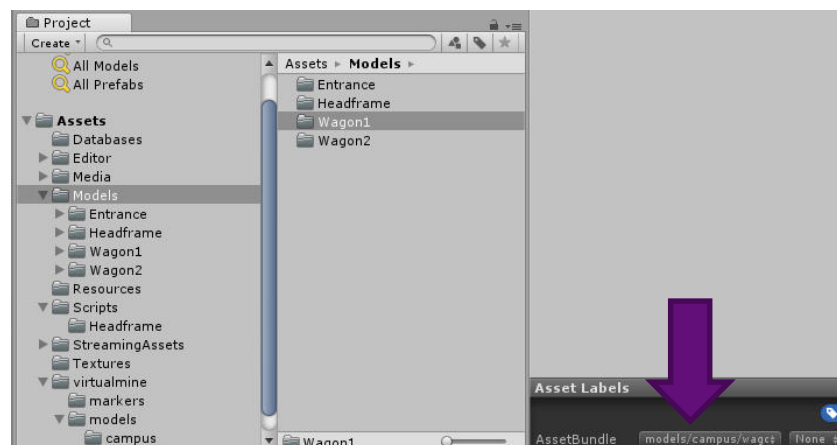
ID (integer)	The ID to identify uniquely this bundle in other tables.
Name (string)	The bundle name code. This value should math [BundleName] in /Models/[LocationName]/[BundleName] for the 3D models and /Media/[BundleName] for the additional media content.
LocationID (integer)	The mine location area ID to be referred in the <i>VMLocation</i> table of the database.

- **MarkerDescription:** this table contains the description for the POIs in different languages

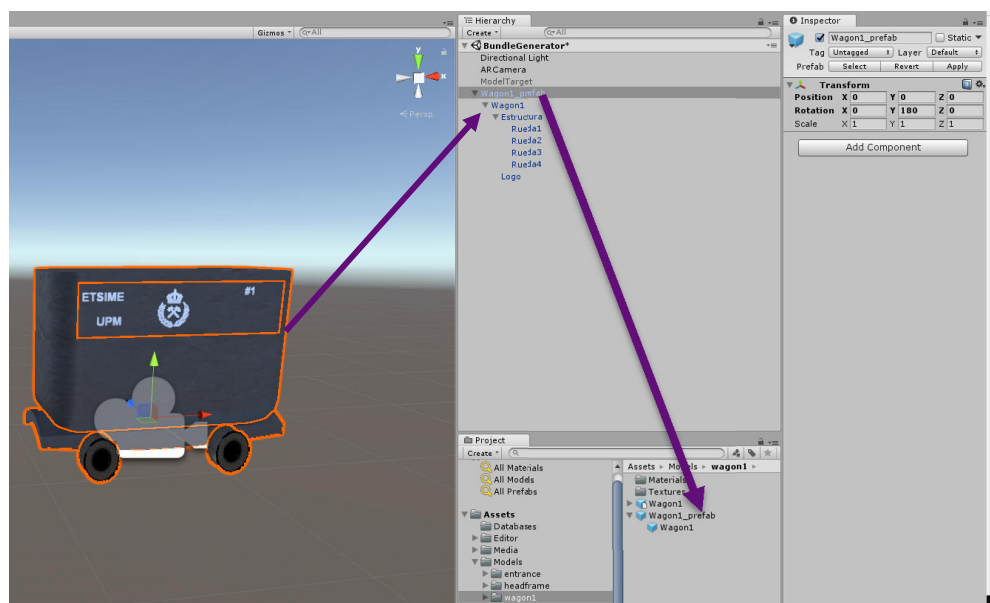
ID (integer)	The ID to identify uniquely this POI description in other tables.
LocationID (integer)	The mine location area ID to be referred in the VMLocation table of the database.
English (string)	The POI description in English language.
Spanish (string)	The POI description in Spanish language.
Polish (string)	The POI description in Polish language.

4.2. 3D models

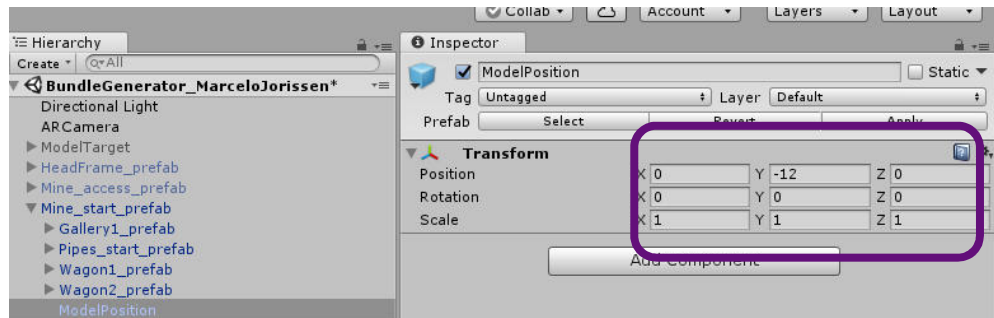
- Create a folder for the 3D model with the name [BundleName] in /Models/[LocationName]/. For example, if we want to create a model in the mine location “campus” with the name “wagon1” we have to create the folder /Models/campus/wagon1
- Create an assetbundle label for the new folder with the name /models/[LocationName]/[BundleName]. In the example, the folder is highlighted in the project view and the label /models/campus/wagon1 is created (orange arrow). To do this, click on the tag -> select New... -> enter label name.



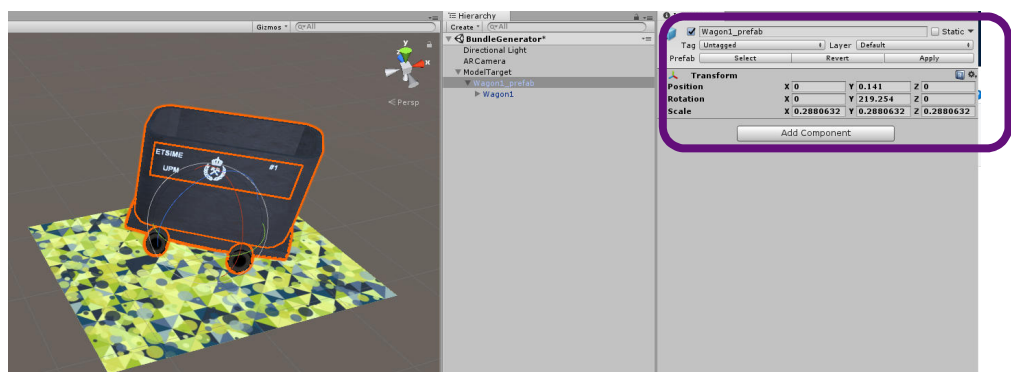
- Import all the content for the 3D model in the recently created folder. This includes geometry files (any format compatible with Unity), Materials (standard materials), Textures and Animations. It is important that all model dependencies are inside this folder in order to download everything from the server when the application is running.
- Create a Unity model prefab with the name “[BundleName]_prefab” and include it in the folder. To do this create an empty object at position zero in the scene, name it accordingly and include the model as child of this object in the Hierarchy view. Before saving the prefab, scale up or down, edit position and rotate the model as desired to show in the real world (consider one unit in Unity is one real world meter and z axis points to the north). When the model prefab is ready, drag it to the created bundle folder in the Project View of the editor to save the prefab (see picture). After saving the prefab, the object can be deleted from the scene.



- Optional: if object in the real world should appear below or over the ground level, create an empty game object in the hierarchy with the name “ModelPosition” and set its position value relative to the parent object. In the figure example, the model is configured to be 12 meters below the ground level.



- Generate Position, Rotation and Scale values for the 3D model that is intended to be shown when the device camera points to the Vuforia AR marker near a POI. To do this, enable the object “ModelTarget” in the hierarchy view and drag the created prefab as child of this object. ModelTarget represents a marker texture (used only for editing) and the goal is to scale, rotate and position the prefab object as you would like it to appear relative to the AR marker (see picture). Write down the values of the resulting transform component (orange square) and fill the field values of the *Marker* table in the SQLite database as described in section 4.1.



- Finally, fill the field *BundleID* of the *Marker* table in the SQLite database with the same value as [BundleName]. In this example “wagon1”.

4.3. Additional media content

Currently, the application supports 2D images to be shown when the user is near a POI in the information panel. To generate additional media content for one POI follow these steps:

- Create a folder for the additional media content with the name [BundleName] in /Media/[BundleName]/. For example, if we want to create media content for a POI related to the wagon model, we can create the folder /Media/ Wagon1_media
- Create an assetbundle label for the new folder with the name /models/[LocationName]/[BundleName]. [LocationName] is the name of the mine location area of the POI. In our example, we create the label /models/campus/wagon1_media. The procedure is the same as described in section 4.2 for 3D models.
- Import all 2D textures inside this folder in any Unity supported format (png, jpeg, etc.). Name these textures alphabetically in the same order as you want them to appear in the application.
- Finally, fill the field MediaBundleID of the Marker table in the SQLite database with the same value as [BundleName]. In our example “wagon1_media”.

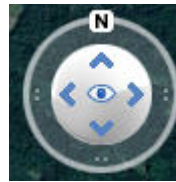
4.4. Map textures

Map textures are used to replace the Vuforia AR map markers with real world textures of the area of interest in each former mine location. The reason for not using these textures directly as AR markers is that quality of detection is very poor for map textures in uniform areas. To configure a map texture, just import the texture in the /Textures/ folder of the project. In the SQLite database, set the value of the field Texture in the table VMLocation to match the name of the imported texture. Anyway, if the map texture has enough quality for detection, it can be used directly as map markers (see section 4.3) and the value of the field *Texture* in the *VMLocation* table of the SQLite database can be set to *NULL*.

Map textures must be obtained using real world applications such as Google Earth. The procedure to create a map texture for a mine location area with Google Earth is described in the following steps (be sure to configure correctly the map texture as otherwise the results of GPS positioning will not be precise):

- Find the area of interest in Google Earth.

- Configure the zoom level to include the whole area, adjust the view angle to face perpendicular to the ground (by clicking the look down arrow in the view controller to the max limit) and finally orient the map to face north (by clicking the N label of the compass). The following image represents the compass and view angle controller.

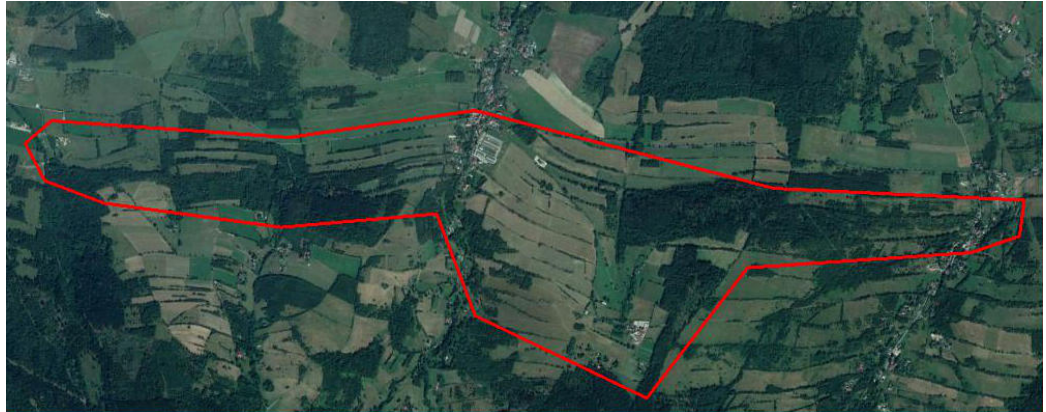


- Create three position markers named BL (bottom-left), BR (bottom-right) and TR (top-right). These markers should be the vertices of a rectangle that include the area of interest. BL and BR should have the same longitude coordinates and BR and TR the same latitude coordinates. You can draw lines between BL-BR and BR-TR to check they are perpendicular and use them as guides to cut the image later. The result should be as the following image, otherwise the view angle and orientation were not configured correctly.



- Write down the latitude and longitude coordinates of the BL and TR markers. These values should be filled in the VMLocation table of the SQLite database as described in section 4.1.

- Hide all the markers and create an image with Google Earth (Edit -> Copy Image or Ctrl+Alt+C). The guide lines BL-BR and BR-TR can be used to cut down the image using an editing program (e.g. Paint). For this example, the result of cutting down the image should be like the following image.



- Import the map texture in the /Textures/ folder if you want to replace the AR map marker used for this mine location or if the texture has enough detection quality, use it directly as an AR marker (see section 4.5 for more details).

4.5. Vuforia AR markers

The application uses an AR marker database for each mine location. Each database should have at least two image targets:

- **map:** this target is used to show the map for the current mine location. It can either be the real map texture created at section 4.4 (if it has enough detection quality) or another image. In the second case, the image must be the same size (in pixels) as the map texture generated.
- **default:** this target is used in the open field to show the content related to the nearest POI (considering the distance to the mobile device GPS coordinates). The application also supports other custom markers different than the “default” marker for a given POI. In this case, just create a new target with the name Marker_[ID] where [ID] identifies uniquely the POI in the *Marker* table of the SQLite database. During runtime, given a POI with ID X, the application checks if it exists a target with name Marker_X in the AR database. Otherwise it will use the *default* marker.

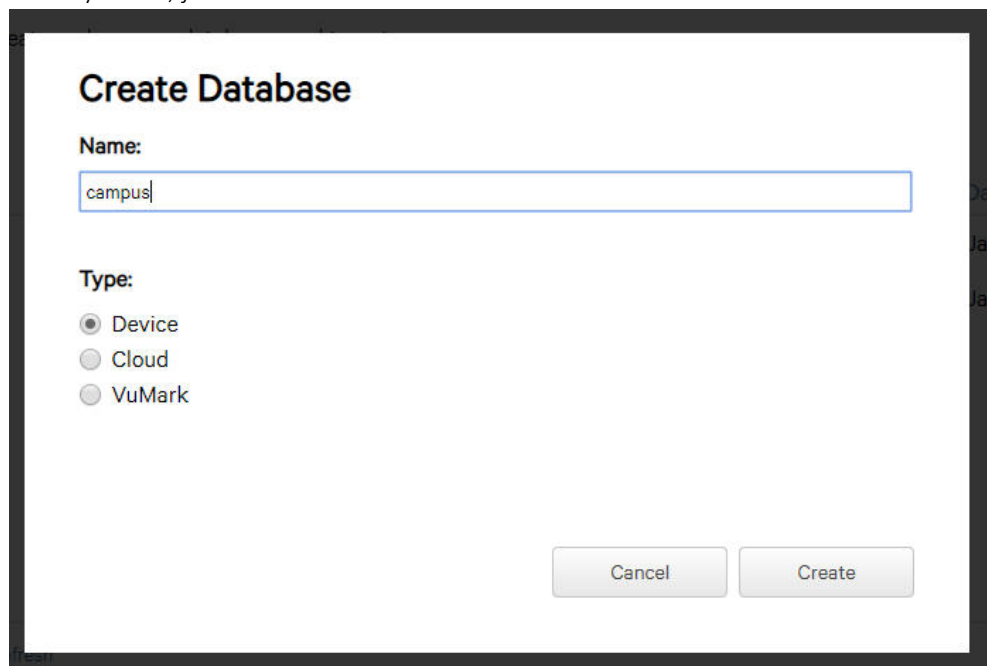
AR databases and markers are created using Vuforia Developer Portal (<https://developer.vuforia.com/vui/auth/login>). For this purpose, a user account has been created for this application:

User: virtualmineft@gmail.com

Password: fastTrack00

To create a Mine Location AR database and markers follow these steps:





- Login to the Vuforia Developer Portal.
- Go to the Target Manager section
- Create a database for the mine location area and name it so that it matches the *Name* field in the *VMLocation* table of the SQLite database (e.g. campus). If the database already exists, just click the name to edit the markers.

A screenshot of the 'Create Database' dialog box in the Vuforia Developer Portal. The dialog has a title bar and a main content area. The title is 'Create Database'. Below the title, there is a 'Name:' label followed by a text input field containing the word 'campus'. Below the name field, there is a 'Type:' label followed by three radio button options: 'Device' (which is selected), 'Cloud', and 'VuMark'. At the bottom right of the dialog, there are two buttons: 'Cancel' and 'Create'.

- Upload image targets using the Add Target button. Browse the image file (jpg format), set the *Width* of the target to 320 and name it accordingly (map, default or Marker[ID]).

Add Target

Type:

Single Image
Cuboid
Cylinder
3D Object

File:

.jpg or .png (max file 2mb)

Width:

Enter the width of your target in scene units. The size of the target should be on the same scale as your augmented virtual content. Vuforia uses meters as the default unit scale. The target's height will be calculated when you upload your image.

Name:

Name must be unique to a database. When a target is detected in your application, this will be reported in the API.

- When uploading the targets, a rating is given to the detection quality. If the rating is below 3 stars, consider changing the image marker. Consider these tips for optimizing target detection:

<https://library.vuforia.com/articles/Solution/Optimizing-Target-Detection-and-Tracking-Stability.html>




- Check that you have uploaded at least a *map* and a *default* target in the database. You should have a list of targets as the following example:

campus [Edit Name](#)
Type: Device

Targets (3)

Add Target

Download Database (All)

<input type="checkbox"/>	Target Name	Type	Rating	Status▼	Date Modified
<input type="checkbox"/>	 Marker_1	Single Image	★★★★★	Active	Jan 04, 2019 18:09
<input type="checkbox"/>	 default	Single Image	★★★★★	Active	Jan 04, 2019 18:09
<input type="checkbox"/>	 map	Single Image	★★★★★	Active	Jan 04, 2019 18:08

- Download the Vuforia AR database by clicking the *Download Database (All)* button. Choose Android Studio, Xcode or Visual Studio as development platform (not Unity editor).

Download Database

3 of 3 active targets will be downloaded

Name:
campus

Select a development platform:

☒ Android Studio, Xcode or Visual Studio
☐ Unity Editor

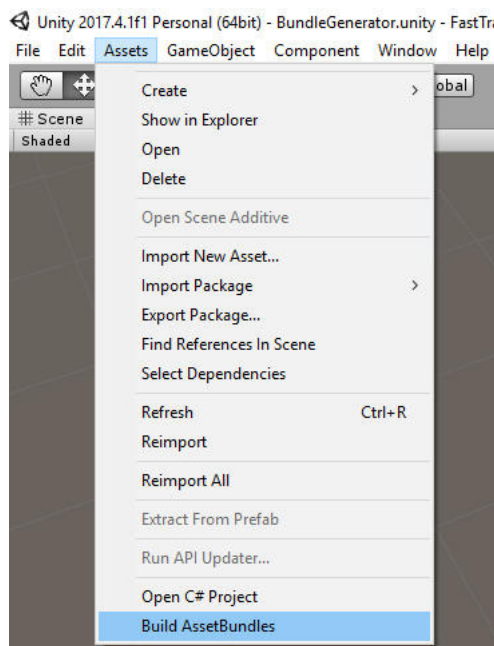
[Cancel](#)
[Download](#)

- The compiled database is a zip file with the name of the AR database. Open the zip file and copy its contents (xml and dat files) to the `/virtualmine/markers/folder` of the FastTrackBundleGenerator Unity3D project.

- Repeat the process for all the mine locations areas configured in the application. Always check that the name of the AR database matches the *Name* field in the *VMLocation* table of the SQLite database.

4.6. AssetBundle generation

This is the last task to do before uploading the application contents to the server. Otherwise, all the changes made will not be taken into account by the application. To generate the AssetBundles, window go to *Assets/Build AssetBundles* in the Unity editor menu (see image). This editor script generates the AssetBundles to be uploaded to the database inside the */virtualmine/* folder. If the bundles already exist, these are overwritten. When the process has finished we can see the result in the console and check if it has been done correctly.

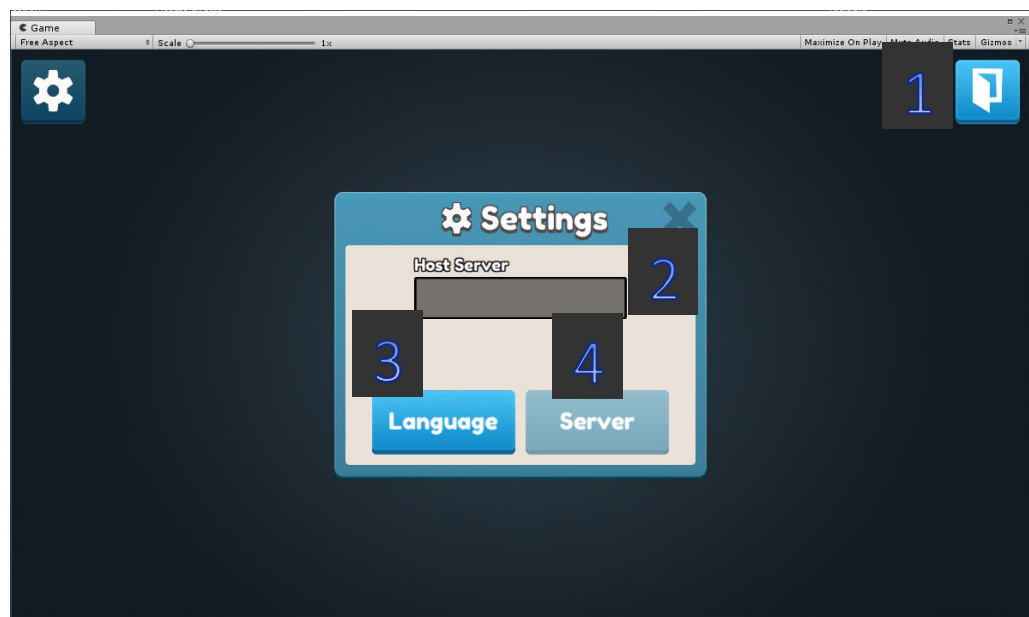


As a result, the folder */virtualmine/* in the project will be updated with all the new contents and is ready to be uploaded to the *virtualmine* host in the server as described in section 3.

5. User interface

This section describes briefly the user interface and functionalities of the application. Consider it as a quick user guide to learn to use the tool.

- A. Settings windows:** When the application starts the settings window appear on the screen. This window enables the user to configure the server address and select the application language. By default, the server panel is shown first.



Other buttons in the settings window interface are described below:

- 1- Exit button: quits the application.
- 2- Refresh/Connect button: tries to connect to the server addressed by the Host Server text input field. As a result, a message appears indicating if the connection has been successful or not. In the first case, the mine location panel is shown on the screen to choose one location (Currently there is a server working at <http://viz2know.cedint.upm.es/virtualmine> but only the server name has to be filled in the input text field e.g. viz2know.cedint.upm.es).
- 3- Language button: shows the language panel where the user can select the current language to use in the application. Last language selected will be remembered for the next execution.

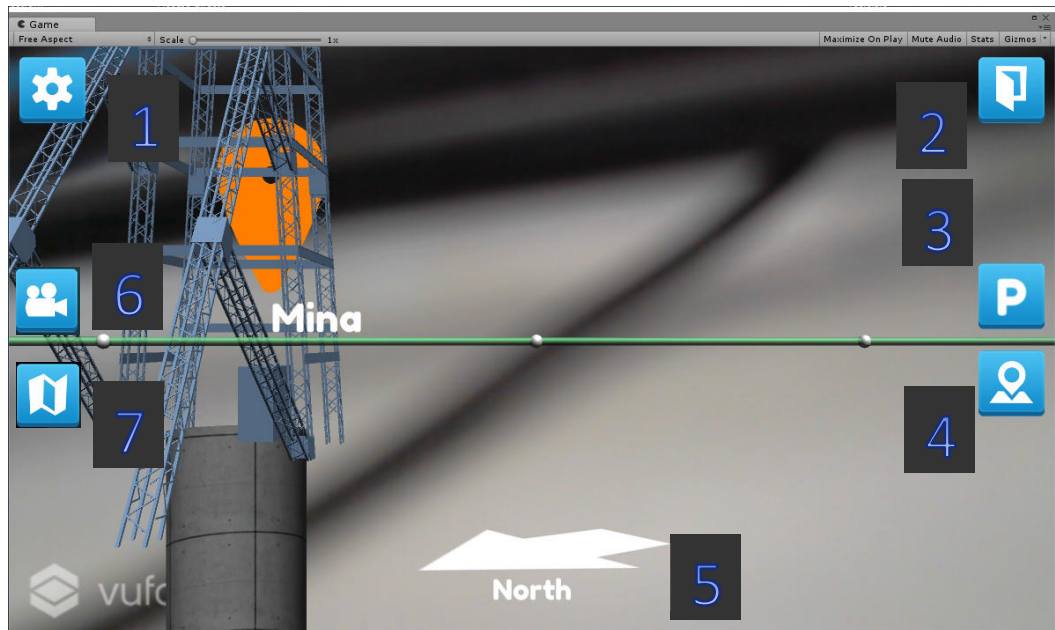


4- Server button: goes back to the server panel previously described.

- B. Mine Location Panel:** when the connection is successful, the user will be asked to select a former mine location area from the list. By clicking the *Confirm* button, all the content and media related to the chosen location will be downloaded from the database.



- C. Navigation Window:** after all the content for the chosen mine location area has been downloaded, the navigation window is displayed. The user is positioned using the device GPS coordinates and the POI's location markers appear at the correct position and distance. By default, the navigation arrow points towards the North to help orient the user in the environment. Whenever a destination is selected (either a Point of Interest or the drawn path), the arrow will point towards the selected direction and the distance to reach the point is shown beneath the arrow. At any moment, the user can select a POI or a path point (white spheres) to travel to that destination.



Other buttons in the navigation window interface are described below:

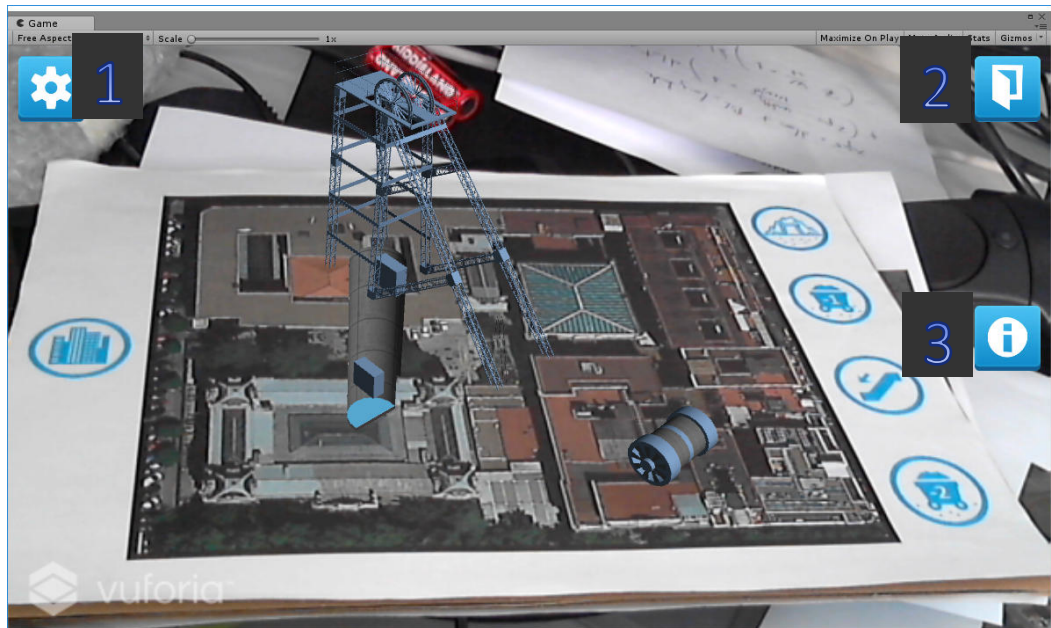
- 1- Settings button: show the settings panel to select/change the current language.
- 2- Exit button: quits the application.
- 3- Go Path button: if the path has been drawn on the map, this button finds the nearest point (white sphere) to reach the path and points the navigation arrow towards that direction. When the point has been reached (device distance below 10 meters by default), the arrow will point to the next point in the path.
- 4- Go Point of Interest button: finds the nearest POI and points the navigation arrow towards that direction. When distance to the POI is below 30 meters, a 3D model appears at the location marker.
- 5- Compass: points to the nearest POI or path point to the user device and shows the distance to that element. If no element has been selected for navigation, the compass points to the magnetic north.
- 6- Look up/down camera button: when selected the device gyroscope is enabled to point upwards or downwards so that objects can be seen over and under the ground level.
- 7- Map button: enables/disables map visualisation from top view perspective without the need of an AR marker.

- D. **Map Window:** when pointing the device camera towards the chosen location AR map marker, the map representation appears and the POIs are positioned at the correct coordinates. User current position is represented with a blue capsule in the map.



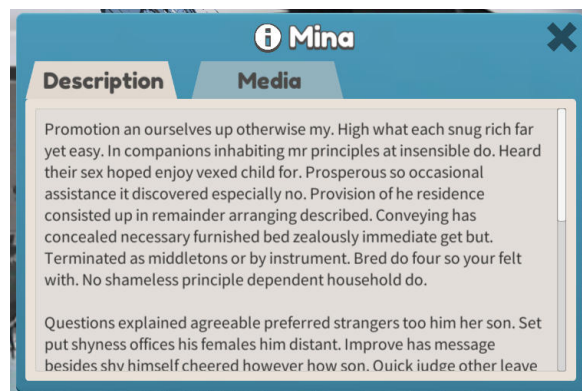
Other buttons in the map window interface are described below:

- 1- Settings button: show the settings panel to select/change the current language.
 - 2- Exit button: quits the application.
 - 3- Draw Path button: when selected the user can draw a path (green line) on the map by touching and dragging a finger on the screen. If a path already exists, this will be deleted and replaced by the new path.
 - 4- Free drawing button: when selected the user can draw any line (red) on the map by touching/holding the finger on the screen. At any time, last drawing can be deleted by making a quick swipe with the finger on the map.
- E. **POI Window:** when the user reaches one POI location and points to the AR marker located nearby, the POI window appears and shows a related 3D model. This model can be either rotated or scaled freely by two finger gestures on the screen. If the current POI does not have a 3D model, the information panel will be shown instead.

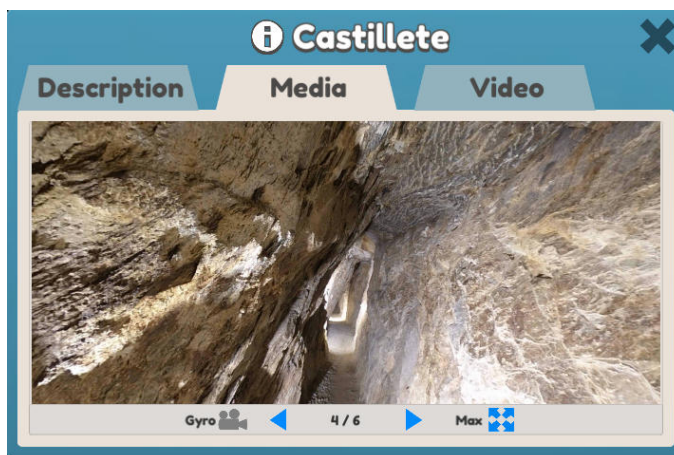


Other buttons in the POI window interface are described below:

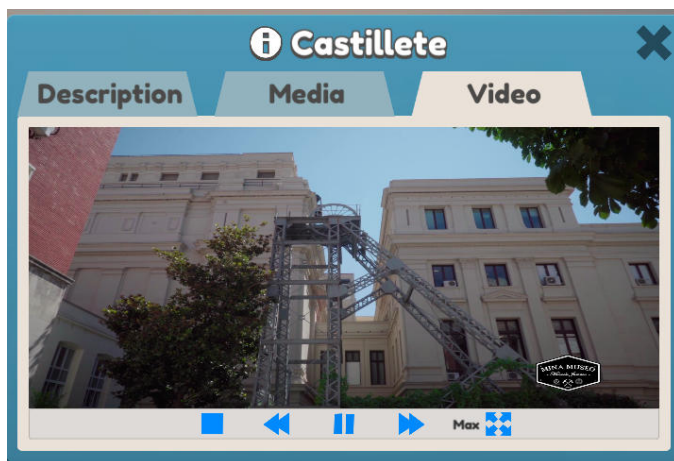
- 1- Settings button: shows the settings panel to select/change the current language.
- 2- Exit button: quits the application.
- 3- Information button: when selected the information panel appears with a description of the POI in the current selected language and other media (photos, 360 photos and videos). If the information panel is enabled, it won't disappear from screen until the button is disabled (being the AR marker present on camera or not). For the media and video tabs there is a fullscreen button to maximize minimize the shown image.



The description tab has information regarding the current POI. A vertical scroll bar is enabled if necessary to read all the text.



The media tab, is a photo gallery with regular photos and 360 photos. In the second case, a gyro button is shown to use the device gyroscope to look around the photograph. Otherwise, the same functionality can be achieved by touching the photo screen.



The video tab has all the controllers to play, pause, stop or rewind/forward the video.

6. Position Simulator

This tool is for development purposes only and it will be removed from the final user version. At the top left corner there is a “Simulator” button with a latitude and a longitude coordinate inputs. When this option is enabled (button turns green) it will offset the device GPS coordinates to these simulated coordinates. This will enable to test the navigation as if the device was in those simulated coordinates.



Some GPS coordinates for the current mine locations configured in the server at CeDInt are the following:

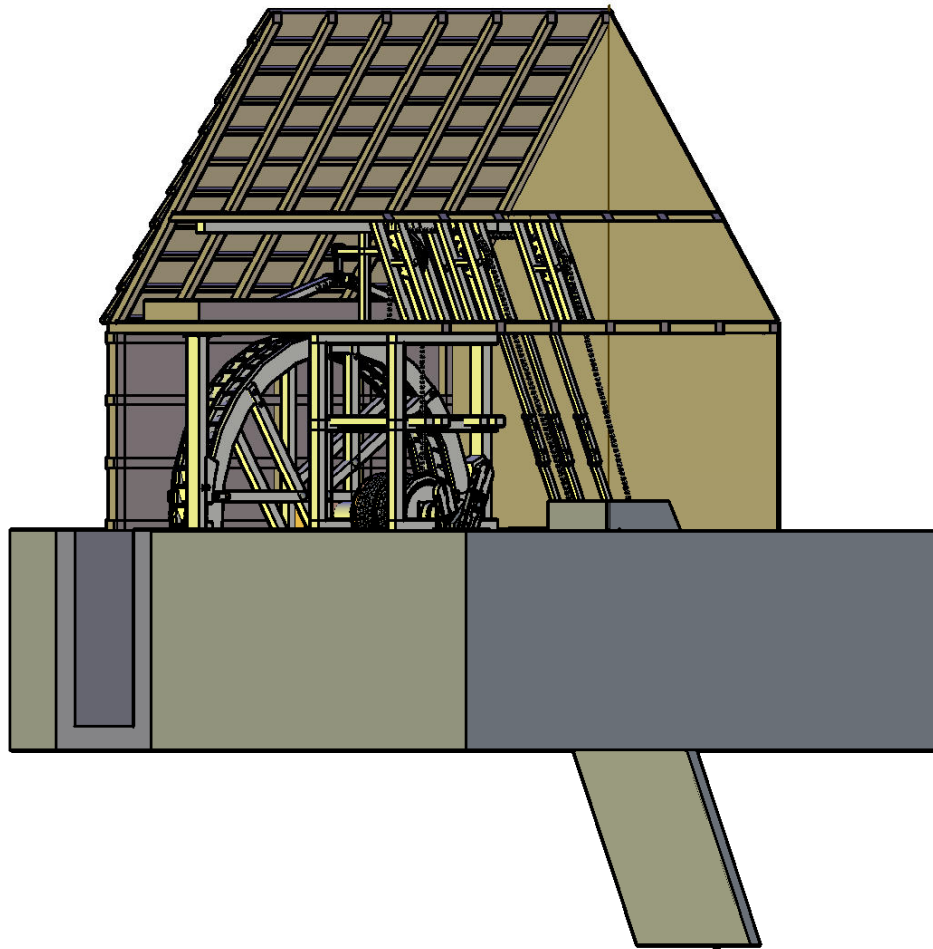
- Campus de Montegancedo (Madrid, Spain):
 - Lat: 40.4041715538052
 - Lon: -3.83438783236626
- Route “By the traces of the former ore mining” (Mirsk Commune, Poland):
 - Lat: 50.928092
 - Lon: 15.380394



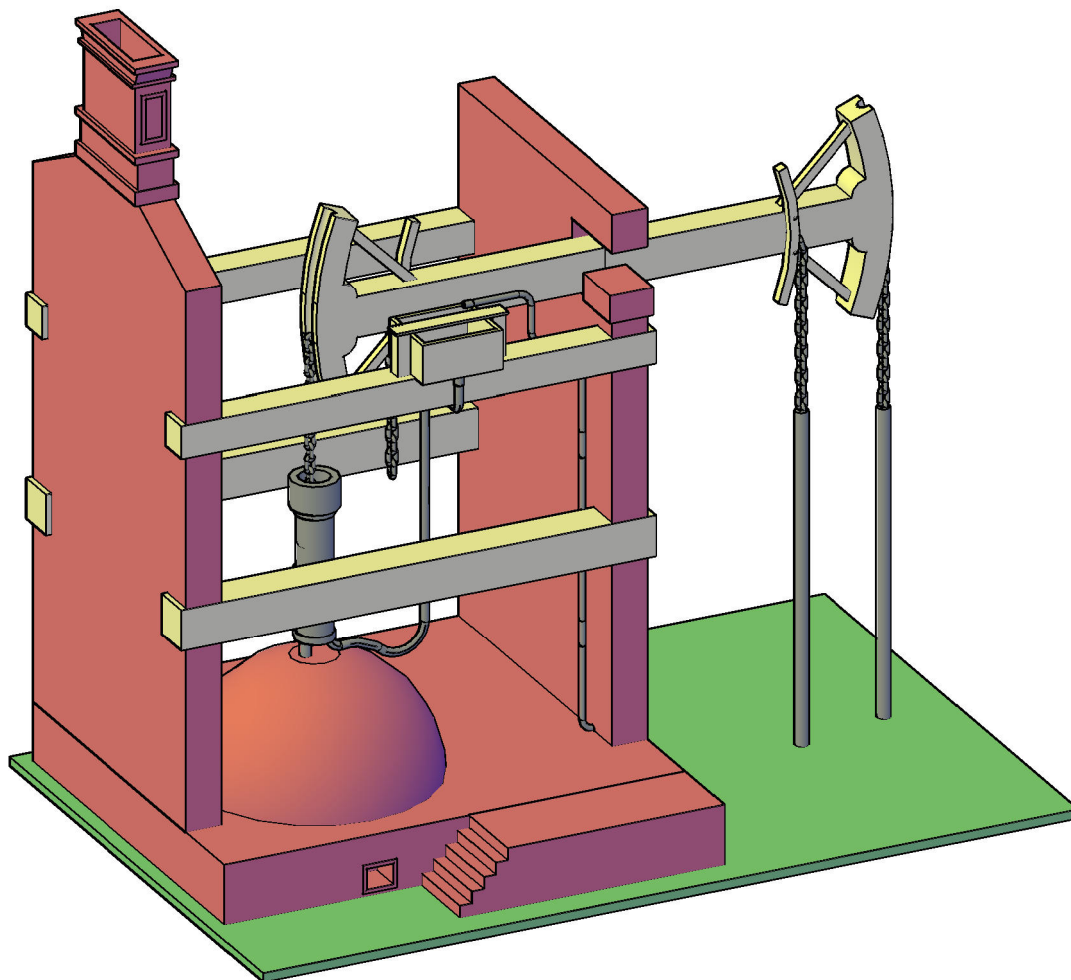
Also, when the map is enabled, a position target button appears next to the simulator coordinates. This button enables the possibility to set the simulated latitude and longitude coordinates by clicking anywhere on the map.



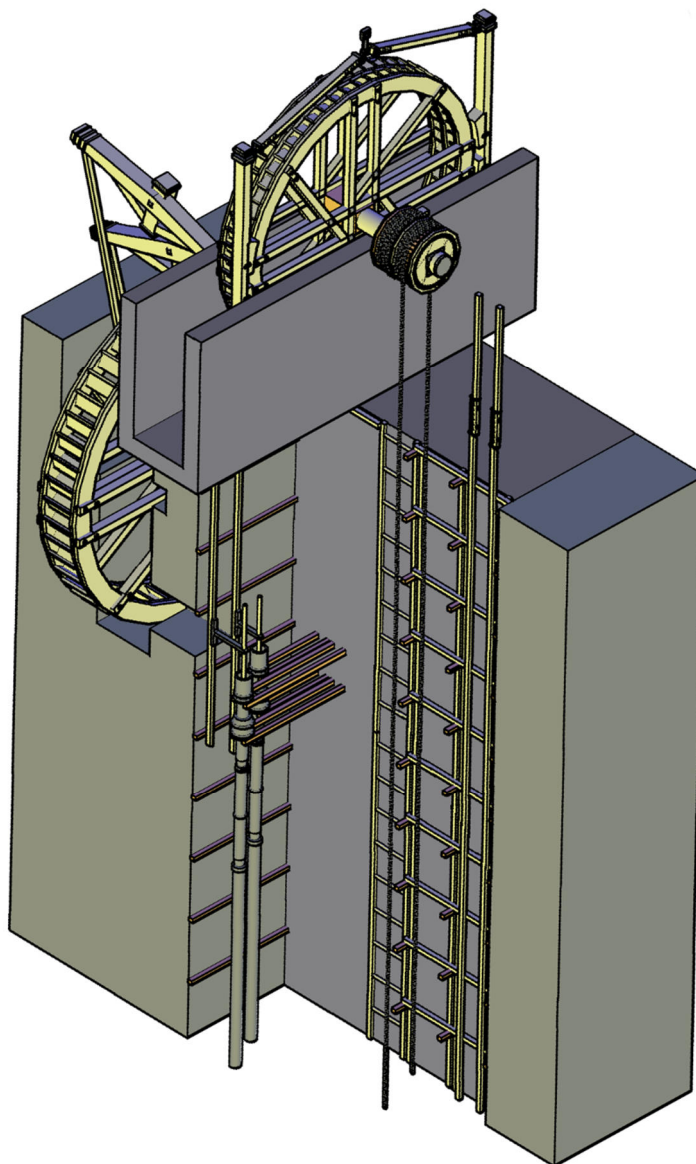
7. Examples of 3D content – selected models



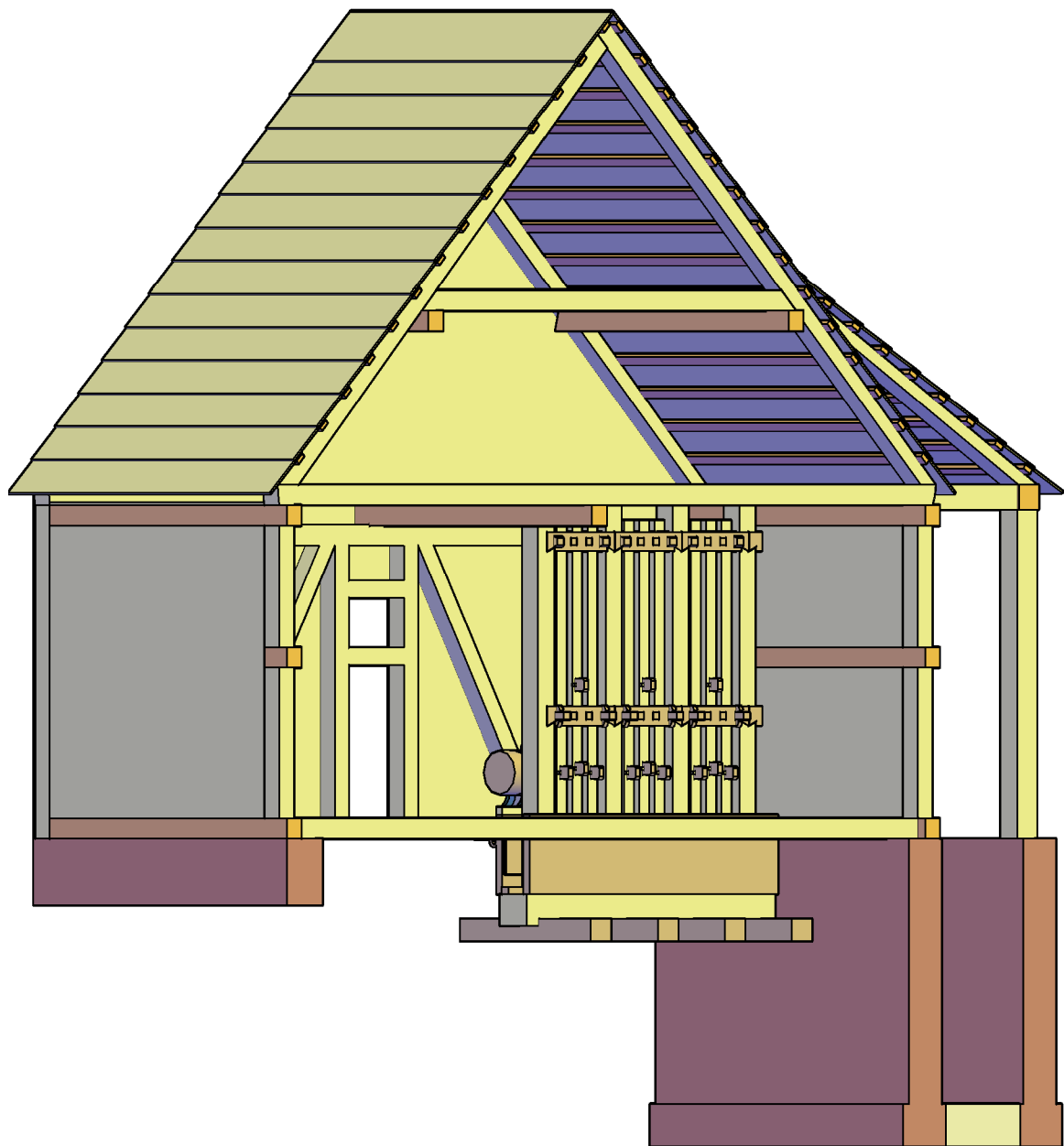
De-watering and lift machine



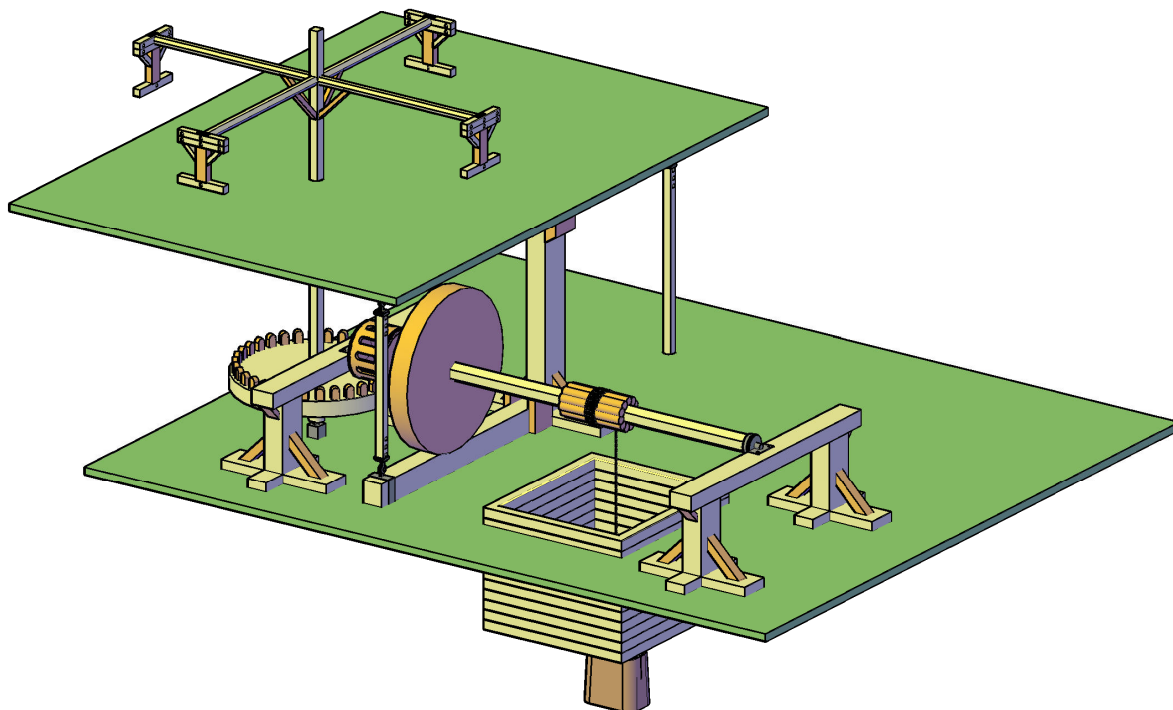
Newcomen's Steam Engine



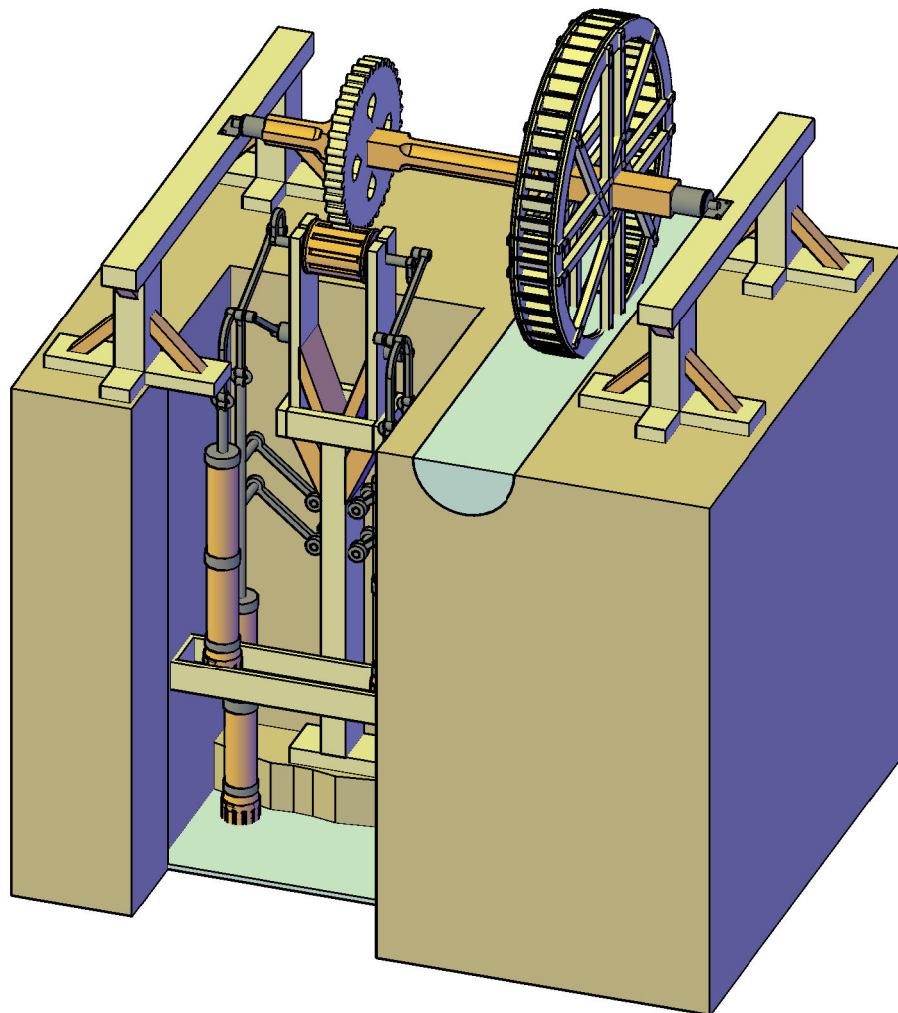
De-watering and lift machine



Old crusher - vibratory pestle



Horse treadmill



Pumping machine - water spinning wheel

