BrickBench

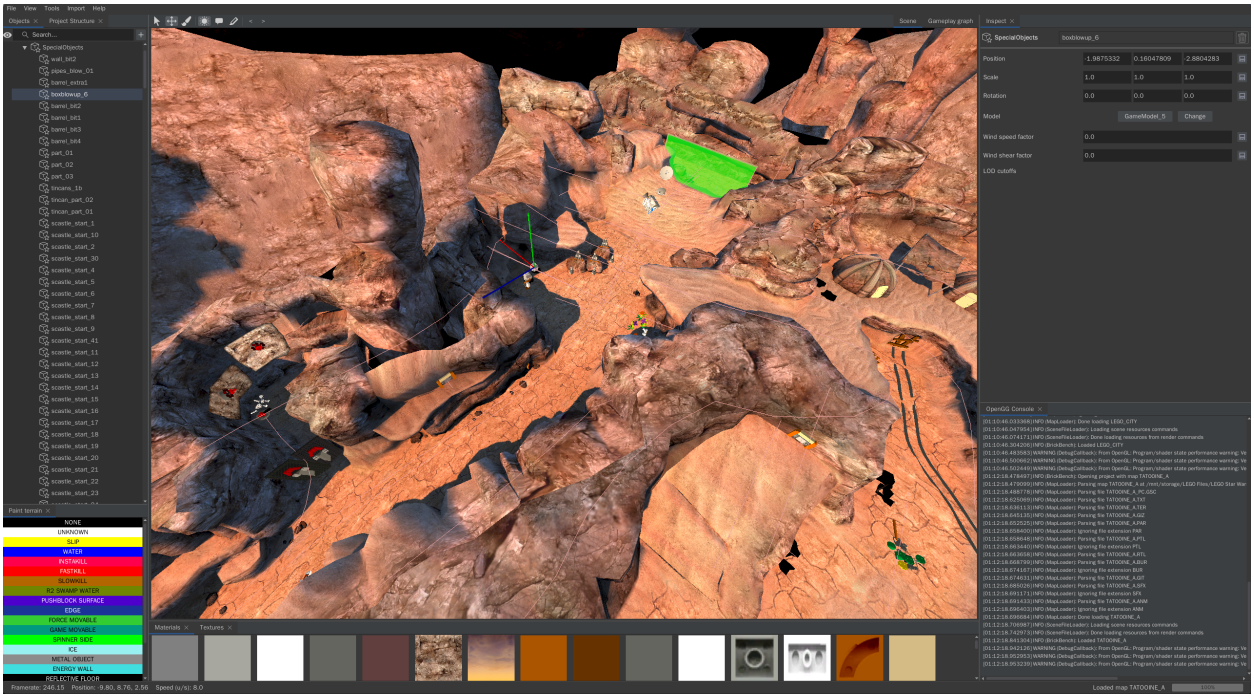
Release 0.3

OpenGG Team

Nov 13, 2021

CONTENTS

BrickBench is a map editor for Lego Star Wars: The Complete Saga.



Note: This project is under active development.

CHAPTER

ONE

CONTENTS

1.1 Setup

1.1.1 Prerequisities

To begin editing map files for Lego Star Wars: The Complete Saga, you must have an extracted copy of the game. If there isn't one yet, please follow this tutorial. If the extraction went properly, your game directory should contain the following folders:

Once you have these folders, delete the `.dat` files in the game directory so the game uses your mods instead of the base game files.

Warning: It is crucial that you have a **clean** extracted install prior to using BrickBench. BrickBench uses existing game files to compress project files considerably, but using an edited copy of the game as an install can permanently corrupt your files and make them unusable by anyone else. If you are unsure if your copy is clean, re-extract a copy for BrickBench to use. Later on you'll have the chance to delete your new clean copy if you want to save space.

1.1.2 Installation

To find the most recent releases, go to the BrickBench GitHub releases page (currently v0.3.0).

There are two ways of installing BrickBench on Windows:

Installer (recommended)

To download the installer version, download `BrickBench-v0.3.0.msi` from the most recent Github release. Once installed, you should be able to run it through a shortcut or as a normal Windows program.

Note: If you install BrickBench this way, you will be able to double-click .brickbench files to automatically open them through Windows.

Advanced: BrickBench can also be run on the command line with `brickbench.exe`. If you pass in a folder or project as an argument, BrickBench will automatically open it.

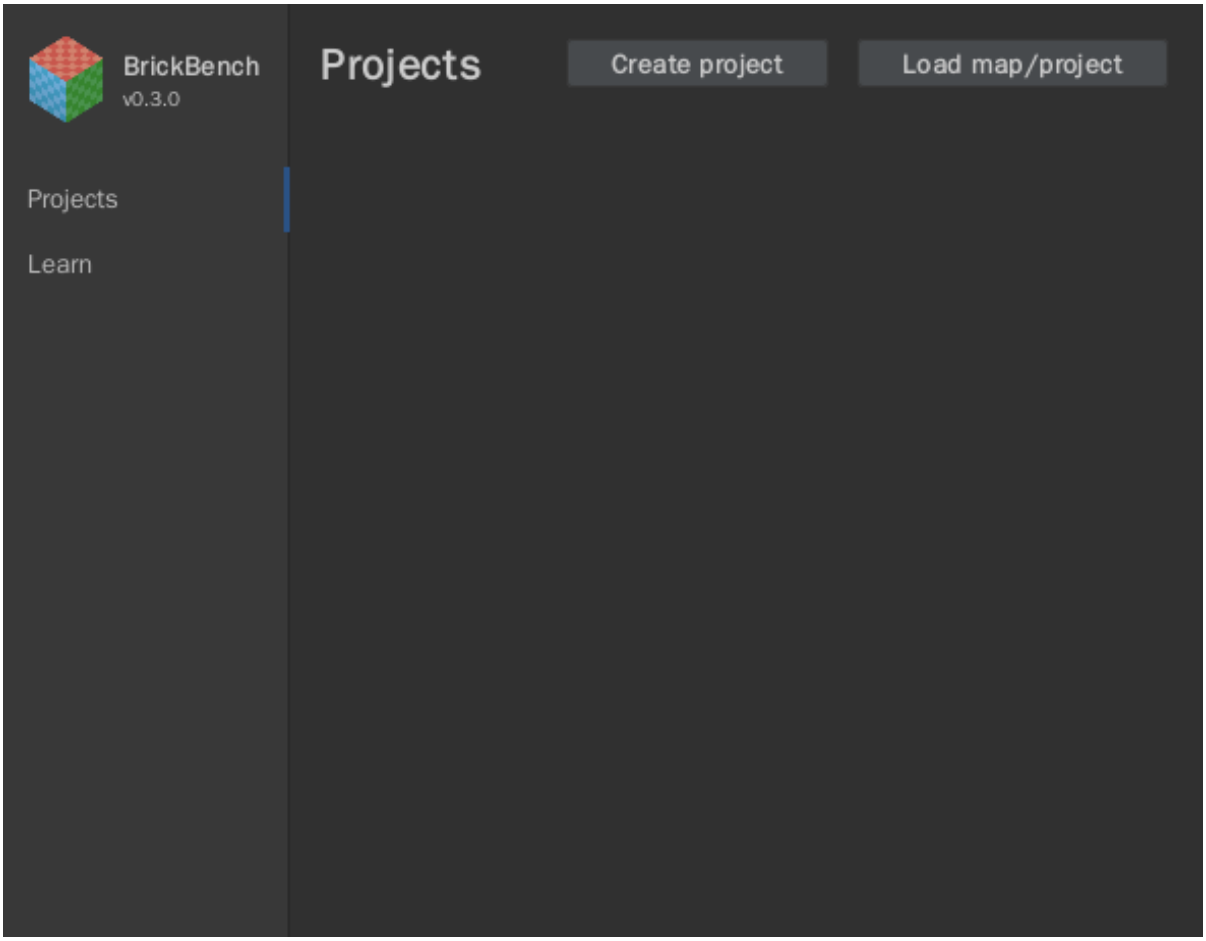
Manual

If you do not want to install BrickBench or do not have administrator permissions, you can download the packaged version. To do so, find the most recent release and download `BrickBench-v0.3.0.zip`.

Once downloaded, unzip it into a directory of your choice. In the install directory, you should find `BrickBench.exe`.

1.1.3 Starting BrickBench

When you run BrickBench, you will be greeted by the following window:



Once started, you have two options: open an existing map to view, or create a project to edit maps.

1.1.4 Viewing maps

To begin exploring existing maps, select the 'Load map/project' button on the Projects tab. Afterwards, select a map folder or a scene (.gsc) file to open it in BrickBench.

After opening, you will notice that the titlebar of BrickBench says Read-only. This means that you are currently viewing a map and cannot edit it. if you would like to edit the map, please create a project and add the map to it.

1.1.5 Creating your first project

To create a new project, click the 'Create Project' button. BrickBench will ask you for a project name (choose whatever you want) and a folder to put the project file into. Once saved, your file will be named `<name>.brickbench`.

Adding maps

At the moment, your project is empty. Let's change that.

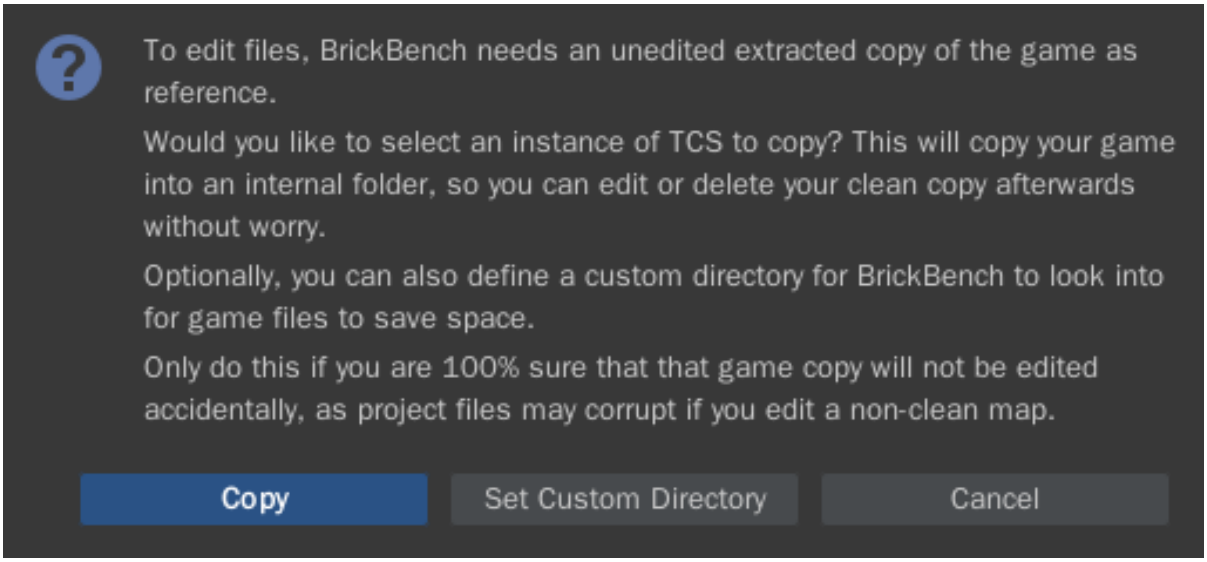
To add a new map to your project, go to File -> Import Map/Area and select a map or an area. This can be done by selecting the directory that contains the map/area you want to add (for example, selecting `Levels/Episode_I/Negotiations` to add the Negotiations level).

Note: The difference between a map and an area might seem confusing at first, so here's what each one means.

Map: a single play area consisting of a scene file and other optional files. An example would be Negotiations_A. A map is normally part of an area, but can also be independent, such as the legal screens.

Area: a collection of maps that are connected in some way through doors and share characters, AI state, and certain other properties. An example would be Negotiations, which contains Negotiations_A, B, C, Intro, and Outro as maps.

If this is the first time you open a map/level, you will see the following prompt:



There are two options:

- Copy: This option copies the game directory you select into an internal BrickBench folder, after which you can delete the directory you selected if you want. This option is highly recommended, as it ensures that there are no accidental changes to your files that may corrupt or prevent loading of projects
- Set Custom Directory: This option sets a custom directory that BrickBench will use as a clean copy. This is useful if you do not have enough space on your main drive.

We *highly* recommend that you allow BrickBench to make a copy of your files. If you make a project using a modded base directory, your project may be corrupted **unrecoverably**.

Once you have your map in, *time to edit*!

1.2 Editing

1.2.1 Editor basics

Warning: BrickBench is still experimental, and currently has **NO UNDO** feature! Please back up any projects you are using constantly, especially when doing non-reversible operations like model importing. You can backup projects by copying the project .brickbench file.

BrickBench editing operates on *entities*, which are objects that can be selected and manipulated on the editor. Entities can belong to a map, in which case they can only be selected when their map is open, or they can belong to a project in which case they're accessible whenever the project is open.

The currently selected object is visible in the Inspector tab, which is on the right by default.

Note: If you cannot find the Inspector tab, it may be closed by accident. You can open closed tools by selecting them in *File->Tools*.

Tools

- **Inspector**: The primary tool for editing. It allows you to edit object properties for all objects.
- **Project Structure**: Allows you to view and edit the project structure.
- **Objects**: Shows you the standard (non-asset) entities belonging to the current map.
- **Materials**: Displays the materials belonging to the currently open map.
- **Textures**: Displays the textures belonging to the currently open map.
- **Paint Terrain**: Allows you to apply custom properties to the terrain mesh.
- **Runtime Hook**: Allows debugging and manipulation of an active game instance.

1.2.2 Editing in the Inspector tab

The Inspector tab is the main workspace for viewing and editing object. It lists object properties, some of which are editable. Common properties are:

- Name: The name of the selected object. In most cases this name should be unique within the given object type (eg. You can have multiple objects named *Object1*, but not multiple gizmos).
- Position: The position of the object in the game world. One common point of confusion is that the engine that TT Games uses a coordinate system that reverses the X coordinate. BrickBench will try to adjust for that when doing things like model importing, but it may be confusing in some cases.
- Rotation: The rotation of the object in degrees, either in one dimension or in all three. Due to how rotation works internally, some objects may act weirdly when rotated 180°. The reason for this is explained in the *advanced* section of this page.

1.2.3 Editing terrain properties

To paint terrain, open the Paint Terrain tool and select the surface property you want. Once you select the property you want, click the painbrush icon in the toolbar on the top of the window. You can then click on terrain meshes to paint them with your selected property.

To finish editing terrain, click the pan/select buttons.

Note: Some surface properties may behave differently on different maps, as the game can redefine how a property behaves depending on the map.

1.2.4 Adding objects

Adding objects is done through the Objects tab. To add a new object, click the + button on the top right, and then select the object you want to add. For most standard objects, this will place your new object onto the current camera location.

Certain, more complex objects (such as models) will select a new object that can be used to tune object properties before adding the object to the map.

1.2.5 Importing assets

Project assets are imported through *Import -> Import Asset*. If you would like to add a new project asset, details are available *here*.

Importing textures

Note: The TT Games engine primarily supports only DDS files (there is basic PNG support). Therefore, BrickBench expects any textures that are added to be in a .dds format. If you do not have your textures in .dds, please convert them before using them either alone or as part of a model.

To import a texture, select the texture in the Textures list in the Import menu and click Next. You will then be able to select if you would like to import the texture as a new texture or to replace one or multiple textures. you can select multiple textures to quickly replace a set of textures with your asset.

Importing models

To import a model, select your model in the Models list in the Import menu and click Next. You will then be able to select the following options:

- Reverse winding: This should be used if your model appears inverted and the terrain colors seem inverted (often times seen as upward-facing surfaces being purple) after importing.
- Mirror on X axis. As noted above, TT Games's engine has an inverted X axis from many editors. If BrickBench does not properly compensate for this, you can force a mirror with this option.
- Generate with normal shading: This enables lighting on your imported model. This should be selected if you would like lights to apply to your model. Note that this option requires your model to have proper normals exported from your 3D editor.

The import process may take some time. After importing, you will be able to see your imported model in the Objects tree. You can then add your model into the map like any other object.

> **Warning:** Model importing is a very complex process internally with many moving parts. In addition, there is no way currently to undo an import operation. Please **BACK UP** your project before doing any model importing.

1.2.6 Advanced

Editing files directly

If you would like to edit a map file directly (either in a hex editor or a text editor), select the map the file is in in the Project Structure tab. Then, in the Inspector, double click the file you would like to edit. This will open the file in your editor of choice.

Once you are done, be sure to save your work for that file BEFORE you save the project.

Internals

Since most file formats aren't completely understood, BrickBench immediately applies any user changes onto the file that is being edited and reloads the file. This allows for great flexibility when editing almost completely unknown file formats, but causes editing to be slow (especially on operations with multiple steps). In addition, this causes certain operations to be confusing (eg. if you rotate something by 180°, it can be ambiguous how that rotation was done so BrickBench chooses a different rotation than what you gave it)

1.3 Project

BrickBench projects are a collection of maps, levels, and resources, and are the primary way to use BrickBench.

1.3.1 Project Structure

BrickBench projects have a set of entities that they manage in what is referred to as a *project structure*. This structure can be seen in the Project Structure tool. It represents the export structure of a project, and is meant to match that of the game in most cases to allow for better compatibility with other mods.

These entities can be edited by selecting them in the Project Structure tool, which will open them in the Inspector.

The project structure consists of the following objects:

Maps

A map in BrickBench refers to a single play space, consisting of a scene file (.gsc) and accompanying files of the same name. They can be contained within an area, in which case the map will be named <area name>_<map ID>, or they can be separate from an area as a loose scene file (as in the legal screens).

Valid map IDs for compatibility are:

- A,B,C,...: These are meant to be used for normal user-playable maps.
- Intro, Outro, Midtro: These are used for cutscene maps.

Maps are exported into either the directory they're in if they're a loose scene file, or under the area directory if they're part of an area (eg. if the Negotiations area is in `Levels/Episode_I/Negotiations`, the Negotiations_A map will be in `Levels/Episode_I/Negotiations/Negotiations_A`).

Note: BrickBench allows the user to manage maps separately from areas even if the map belongs to an area. We recommend against this in most cases, as this will prevent you from adding AI characters properly as that information is stored as part of the area, not the map.

Areas

Areas are a collection of maps that share some properties, such as AI state. Their properties are defined in the *level TXT*, which is the .txt file in the directory the level is in (so `Negotiations.txt` in the `Negotiations` folder), and in the `AREAS.TXT` file, which is in the `Levels` folder.

Resources

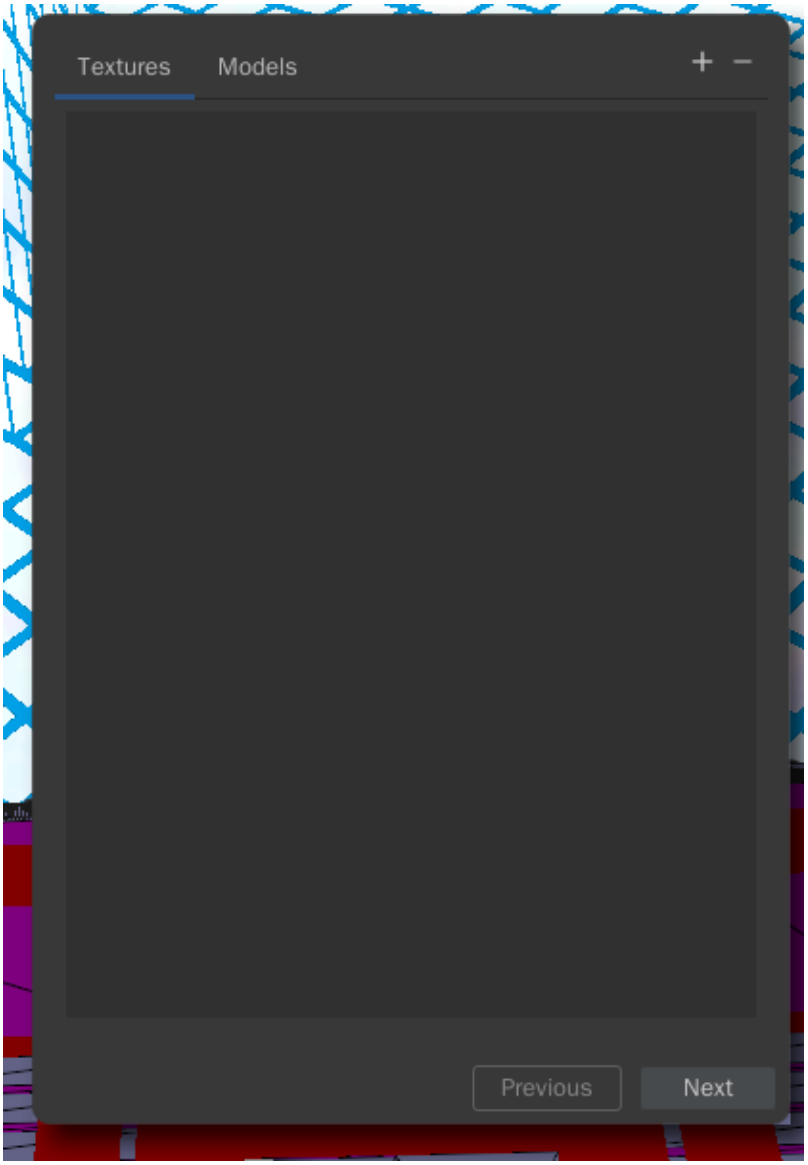
Resources are any files or folders that may be of use for a map or area to work properly. Some examples are music (.ogg), custom ENGLISH.TXT files, and characters. These can be added to the project structure and will be exported along with the project.

Note: BrickBench **CANNOT** edit these files, and no support for editing non-map files is planned. Resources are meant to be used solely as a convenient method of packing files in with your project.

1.3.2 Assets

Projects also store a number of *assets*, which are models/textures that can be used in maps. They are stored separately from maps to allow for easy reuse of assets in different maps, even if you are editing someone else's project and do not have the assets on your computer.

Assets can be imported through the Asset Import window, which can be selected by clicking Import -> Import Asset in the menubar.



Click the + button to add a new asset. Assets can be either textures (.dds) or models. BrickBench currently supports .obj and .fbx models. To import these assets into your map, see *here*.

1.3.3 Exporting

Map exporting is done through the Export menu under File->Export. Projects can be exported either as individual maps or as a whole project, in which case each project entity will be exported as per the project structure.

If exporting as a map, there are some additional auxiliary files that you may want to export to allow your project to load properly. These files are the `LEVELS.TXT` and `AREAS.TXT` files, which define what maps and areas are loaded and how. There are three different export methods for these files:

- None: These files will not be exported.
- Stub: These files will be exported, but only containing any changes from the original files. They will be exported as <file name>_<project name>.TXT (eg. LEVELS_MyProject.TXT). This is the recommended export method for distributing mods, as it allows the user to more easily merge your mods into their game.
- Full: These files will be exported in their entirety, including both the original contents and your modded contents. This makes it easier to test your project (although we recommend that you test your project *like this*). However, please do not distribute your mods like this, as it makes it much more complicated to merge two different mods together.

1.4 Testing

BrickBench supports a set of features tailored to testing and exploring maps.

1.4.1 Running your project

The recommended way of testing your project is to run a test instance, done with *File->Test Project*. This will create a new game copy, copy your project into the copy, and run it. This is preferred over exporting as it allows you to not have to manage a modded copy of the game. It also prevents issues like .pak files not getting removed.

Note: BrickBench will create a copy of the game next to your clean copy. This copy will take up **NO** extra space on your computer (other than the space used by your project files). BrickBench should delete this copy once you close your game, but if it fails to do so it is safe to delete it manually.

Warning: BrickBench uses *hard links* to create the test copy of the game. Therefore, **DO NOT** manually edit the test copy the game makes, as those changes will be reflected in your clean copy. If you would like to change files in your test copy, add them as resources as explained *here*. This is both safer and allows your changes to travel with your project.

Deleting files does not count as editing, so that is safe to do.

1.4.2 Runtime hook

BrickBench supports a runtime hook to allow viewing and editing of certain game properties while the game is running.

To open a hook, open the Runtime Hook tool and click the Start Hook button while the game is running. This will open a connection to the game and allow BrickBench to read memory from the game.

Once the hook opens, you should be able to see player 1 and 2 as blue and green rectangles respectively, and any other characters as yellow rectangles. You can use the *T* key to teleport player 1 to the current camera position, and *Ctrl+T* to teleport player 2.

In addition, BrickBench allows you to quickly teleport to any map in the game. To do so, select the map in the dropdown in the Runtime Hook tool and click *Load Map*. This will reload the map when you return focus to the game.

Note: If the dropdown isn't sending you to the right level, you can also manually set the map ID to load.

Hook options

The runtime hook has the following options: * Autoload map from current hooked game: When this option is enabled, BrickBench will

automatically load the map the player is in.

- Reset map on load: This option resets the map state when a map is loaded.
- Reset door on load: This option resets the door that the player will enter through when entering a map directly.
- Enable global hotkeys: This enables the teleport hotkeys while not in BrickBench.

1.5 Glossary

1.5.1 Object types

Scene objects (.gsc)

- Splines: 3D lines in space that can be used for various things. Some common ones are:
  - Camera splines: Splines that are used to define camera movement (the camera properties are configured in the .txt file).
  - Spawn splines: Splines used to define spawn points for characters, often named start.
  - Door splines: Splines that define a doors' bounds in addition to a spawn spline for when that door is exited from.
- Mesh: A piece of a 3D model with a fixed vertex format.
- Material: Defines a shader configuration and shader parameters to render a mesh.
- Model: A complete 3D model, combining a set of meshes with their corresponding materials.
- Model instance: An instance of a model placed in the map.
- DisplayList: A list of model instances, plus other render parameters.
- SpecialObject: A model instance that can be moved and/or animated. This is often the base for all interactable objects ingame.

Terrain objects (.ter)

- Terrain Group: A mesh consisting of lists of quads with optional surface properties, used for collision.
- Terrain Platform: A terrain group attached to a SpecialObject, used for moveable terrain.
- Infinite Wall: a list of points defining the borders of a 2d vertically infinite wall.

AI (.ai2)

- CreatureSpawn: A location at which an AI character may spawn when the map is loaded. The character type and active script are defined in the creature spawn.
- Locator: A location that can be used by AI in scripts as a marker or destination.
- Trigger: A zone in 3D space that can be used by scripts to see if a specific character is within the zone.

Gizmos (.giz)

- Pickups: Items that the player can pick up by touching/approaching them (ex. studs, minikits, hearts).
- Lever: Pullable levers.
- Panel: Panels that only allow certain character types (or characters disguised as those character types) to interact with them.
- HatMachine: Objects that allow most characters to get a specific set of headwear, often for use with Panels.
- ZipUp: Objects that blaster characters can zip/swing around with.
- PushBlock: Pushable blocks, often for use in puzzles.
- Gameplay Graph (.git file): Defines connections that allow gizmos (and other objects) to cause changes in the game world.
  - An example would be a lever causing an obstacle to move.

1.6 FAQ

1.6.1 Troubleshooting

Editor controls

I can't select any objects

If you cannot select objects, make sure that you have selection mode enabled in the toolbar at the top of the screen.

I selected a terrain type for editing, but clicking on terrain doesn't do anythin

Ensure that you are in paint mode, which is selected in the toolbar at the top of the screen.

Model importing

BrickBench says that my model has too many vertices

TT Games uses a model format that only supports 65,000 vertices per mesh. If your model exceeds this, there are two options:

- Simplify the model, removing unneeded vertices
- Split the model into multiple parts, each of which has less than 65,000 vertices.

BrickBench says that my model is not supported on the current map

Certain maps do not have the correct buffer sizes for the BrickBench model format. We plan on changing this in a future release.

BrickBench says that my model contains non-DDS images

TT Games's engine, and BrickBench by extension, only support DDS images. BrickBench will convert the filenames if your material has a texture with a .png extension but the same texture with .dds exists, but it cannot actually convert the file type.

But why can it not just automatically convert my images?

The DDS format has different internal formats (DXT1, DXT3, DXT5), each of which has different features. BrickBench cannot know which features you would like on your texture, and therefore cannot automatically convert them.

Exported maps

I added a bunch of X gizmo, but they don't show up

Gizmos have a specific maximum that is set in the maps' .txt file. If you want to add more gizmos of a certain type than that limit, you need to increase that maximum in the .txt. Editing map files directly can be done as explained *here*

I added a creature spawn to my map with a certain creature type, and it won't show up

If you are adding a creature to a map, its area needs to have that character as resident. That can be done by selecting the area in the Project Structure tool and editing the character list in Inspector.

None of my AI changes are taking effect

If you are exporting your changes to a new directory, be sure to delete the `ai.pak` file in the maps you export into. That file contains a copy of all AI data, and not removing it will make the game use the unmodified AI data instead of your modded copy.

My custom music isn't playing

If you have custom music, be sure that it is in .ogg (Ogg Vorbis) format with 2.0 channels and the bitrate/frequency is similar to other songs in the `Audio/_MUSIC` folder. If you are adding new music, you will also need to override the existing `music.cfg` file. You can do that by adding the replacement file as a resource.