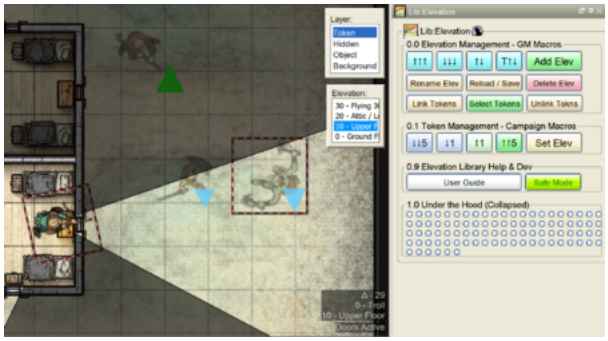
Lib:Elevation Manual

From RPTools Wiki
Jump to navigationJump to search

Contents



Lib:Elevation tries to fit into the MapTool UI to add accessible elevation tools for map-making.

Welcome to Lib:Elevation!

Quick Links

- **Download Lib:Elevation from GitHub (https://github.com/melek/lib_elevation)**
- View the RPTools Forum Post (https://forums.rptools.net/viewtopic.php?f=8&t=29234)
- Ask questions and discuss in the MapTool Discord (https://discord.gg/hbn2bfn) in the #map-drawing-tools channel

Introduction

This library adds an elevation feature to MapTool, allowing you to **set Visual and Movement blocking layers based on elevation** and **set maps, doors, and other tokens for an elevation**. Additionally, player or NPC tokens with an Elevation property can automattically view the elevation they are at when it is their initiative turn.

Here are some ideas on how to use this feature:

- Set up outdoor maps where climbing to a higher elevation removes vision barriers from lower elevations.

- Remove vision entirely to simulate flying or scrying.
- Change the background map at different elevations for a truly 'vertical' feeling, such as flying islands or tall, fog-filled canyons.

Terminology

There are a few concepts in Lib:Elevation that have special names that knowing up front can help you get to using the library quickly.

Some of the terms are similar to other MapTool terms like layer and token, so to (hopefully) reduce confusion, **Elevations** are sets

- **Token Elevation**, or the elevation token property, is the vertical position of a token - how far up or down it is relative to other tokens. This is usually stored on the token as 'Elevaton' (that is the MapTool default), and by itself is just a handy reference for players and GMs to know how high or low a token is on the map.
- **Map Elevations**, defined elevations, or simply elevations, are sets of data - a label, blocking layers, and linked tokens - which is shown at a particular elevation on the map. For example, a map of a maze might have 'Elevation 0' labelled as 'Ground Level' and block movement and sight the walls, while Elevation 20 might be labelled 'Flying' and allow free vision and movement across the whole map. When I use the word 'Elevation', these special, defined elevations are usually what I mean, rather than the token property. Sometimes I may refer to elevations as 'layers' or 'elevation layers', but I try to reserve that term specifically for the VBL/MBL associated with an elevation.
- **Current Map Elevation**, current elevation, or active elevation is the map elevation being displayed. Only one elevation is visible at a time for all players. Switching elevations means changing the current map elevation, which literally 'switches out' which elevation you are playing on/editing.
- **Faded tokens** are tokens on a defined map elevation above or below the current elevation on the Token layer. These tokens will appear 'faded' with lower opacity and placed behind tokens on the active elevation, and when they are defined will have the `isAbove` or `isBelow` state applied to it.
- **Elevation Linked Tokens** or **Elevation Tokens** refers to tokens which have their positions 'linked' to a particular map elevation. These let you do thing slike switch the background map token, only show an NPC when they are at the same elevation as the player, hide door tokens so they don't appear at the wrong elevations. If the state is defined, these tokens will have a GM-only `isElevationToken` state applied to them to make them easier to spot for the GM.

Campaign Setup

Note that the library is designed to (mostly) work without any of the items below defined, but they should be added for you to benefit from the features of the library.

- **Drag the colored elevation macros to your Campaign/GM panels** for quick access. The overlay UI has most of these options, but it is nice to have the buttons handy - especially in case the overlay has to be shut off for some reason. Exporting then importing these macro groups is a quick way to do this.
- **To show token elevation on mouse-over stat sheets**, add a campaign property for '*Elevation'. This is a default MapTool property. If you'd like to use a different property name for elevation in your campaign, you can change it in the `getElevationPropName` function.
- **Add elevation token states in Campaign Properties**, which will be used to show when a token is at an elevation above or below the current map elevation. Add two states: `isAbove` to mark tokens on a higher defined map elevation level, and `isBelow` for tokens below the current map elevation level. Here are recommended settings for these states:
  - Choose the 2x2 Grid shapes at 50% opacity
  - Use 'Yield' (down arrow) for `isBelow`
  - User 'Triangle' (up arrow) for `isAbove`
  - Tokens will also appear faded when not on the current elevation.

- **Add a GM-only state to help manage Elevation Linked Tokens** such as doors, background maps, etc.. Add one state: `isElevationToken` with the following state properties:
  - Choose the 'Shaded' type state set to a bright color and 25% opacity.
  - Check the 'GM Only' option.
  - Check the 'Mouseover' option.
  - You will learn more about why this state is helpful and what these linked 'Elevation Tokens' are for.

Usage

Lib:Elevation works by saving each elevation as a set of blocking layers (VBL and MBL) for the map, along with a list of any linked tokens which will be hidden when a different elevation is loaded. Changing between elevations swaps out which blocking layers are active and which elevation tokens are visible.

By default, new maps have no defined elevations and won't show the elevation list overlay for the GM. These maps are considered to be at elevation zero. You can add an elevation with the `Add Elev` button.

Once two or more elevations are present, a selection list will appear so the GM can easily switch elevations. **Changing elevations affects all connected players**, unlike maps which the GM can freely preview privately without impacting which map a given player is on. If you want to check elevations on a map, it is recommended to keep the map hidden from players until you are ready for them to play on it.

When you switch between elevations, your changes to VBL/MBL and any elevation linked tokens will be saved. Then, VBL/MBL will be cleared, elevation tokens will be hidden, and the elevation you are switching to will be loaded onto the map.

Switching elevations is done manually, or when Initiative is active, when a token takes initiative. You can also use the `T↑↓` button to match the current elevation to the selected token's elevation property.

Elevation Management Buttons

There are some additional buttons, and you can check the tooltips of any button for more information. Here is a rundown:

- `Add Elev` will add a new defined elevation. You can either start with the same MBL/VBL that the current elevation has, or start with a fresh slate. Similarly, you can copy the elevation tokens from the current elevation, hide those tokens (like you would when switching), or leave them unlinked to the new elevation but still visible - useful if you want to link only some of them, like a background map token.
- `↑↑↑`, `↓↓↓`, and `↑↓` switches the map elevation up, down, or to a map elevation you choose.
- `T↑↓` button switches the map elevation to match the selected token's elevation property, rounded down to the nearest defined elevation. If a token's elevation is below the lowest defined elevation, it will default to the lowest available defined map elevation.
- `Link Tokens` links the selected token to an elevation you select. You can use `Unlink Tokns` to undo this, or `Select Tokens` to select all the current map elevation's linked tokens. These options are mostly for setting maps up, but you might unlink/link a token during play if moving NPCs between levels in a house, for instance.

Token Management Buttons

These buttons simply let you change the elevation of the selected token(s). Hovering over them provides a quick view of the change they will effect.

Please note that using these buttons usually also switches to the new elevation.

Limitations

There are of course limitations given the current state of MapTool and the limits of my development time.

- Hard fog of war is exposed for all elevations. This limits usefulness for indoor areas, as the fog for one level may greatly impact vision of another area.
- While tokens can have their own elevation, the map can only display one elevation at a time. That means that all your players stuck in a maze will see what the Flying character is seeing on their turn.

For any questions, please go to the MapTool discord and ping Melek in #map-drawing-tools.

Developer notes

In Lib:Elevation, map elevations are special objects set at an elevation number defining walls and other visual/movement barriers at that elevation and higher (until the next defined map elevation). For instance, you may want three elevations 10 feet apart - Low ground (0), High ground (10), and Flying (20+). Each elevation should be considered to 'fill' the vertical space until the next elevation.

Here are outlines the data structures and basic mechanism behind the library.

Library Properties

The library itself contains no properties by default. There is a library options system which lets you set library-wide options, but that feature is not currently in use. Future versions may use these options as a fallback for undefined map-level options.

Note that versions 1.0a4 and lower stored all elevation data on the library token, using map IDs as keys. This was not import/export friendly and was replaced with library reference objects stored on automatically created map elevation data tokens.

Map Data Token properties

On nearly any elevation operation, a 'Map Elevation Data' token will be automatically created and populated with library reference data. All elevation related data on the map is stored on this token. **If you delete the Map Elevation Data token, you will lose all your elevation data for the map.**

Using a token in this manner allows maps to be exported and imported to other campaigns using a sufficient version of Lib:Elevation.

Property	Type	Purpose
libRefData.elevation.version	String	The version of Lib:Elevation which created the data token.
libRefData.elevation.options	JSON Object	Option names are paired with their values. If they don't exist, they are presumed to be 0.
libRefData.elevation.current	String (Integer)	The current elevation. This value is set when saving an elevation or loading an elevation, which happens whenever the elevation is switched with a UI macro. Essentially, this is almost always the 'display' elevation, unless you have manually used macros to load different elevation layers or hide/load elevation tokens from different elevations.
libRefData.elevation.layers	JSON Object	Integer IDs representing an elevation are keys for JSON objects containing elevation data. This data structure is described below.

Data Structures

Note that Lib:Elevation uses Map Data Tokens stored on each map; this means that the library is loosely coupled with a campaign, and maps can be freely exported. Token IDs should stay constant on map import, so token names and exports/imports shouldn't compromise elevation data.

`libRefData.elevation.layers`

This JSON object uses Map IDs as keys to a JSON object containing elevation data, called 'Map Elevation Objects'. Each Map Elevation Object contains numeric keys designating the elevation it describes. Each elevation in turn contains an object with the following optional keys - no keys are mandatory, though omitting some keys will define default behavior:

Key	Type	Purpose
label	String	A display name for the elevation. Defined elevations don't require names.
vbl	JSON Array	VBL is a shape object array as returned by `getVBL()`.
mbl	JSON Array	MBL is a shape object array as returned by `getMBL()`.
tokens	JSON Object	Token ID keys containing **token appearance** objects. Token appearance is described below.

Token appearance objects

Token appearance are the exact parameters of how a token appears on the map. These keys are used in `getTokenAppearance()` and `setTokenAppearance()`. The token appearance JSON Object includes the following keys and the associated token data:

Key	Data function
ID	Same as Field ID
Layer	`getLayer(ID)`
X	`getTokenX(1, ID)`
Y	`getTokenY(1, ID)`
Z	`getTokenDrawOrder(ID)`
Height	`getTokenHeight(ID)`
Width	`getTokenWidth(ID)`
Size	`getSize(ID)`
Facing	`getTokenFacing(ID)`
Opacity	`getTokenOpacity(ID)`
Visible	`getVisible(ID)`
Shape	`getTokenShape(ID)`
Layout	`getTokenLayoutProps(';', ID)`

Token Fade Data Lists

Tokens on other elevations are faded with the `elevation.setTokenElevationStates` function have their opacity reduced and are assigned either the `isAbove` or `isBelow` state. Additionally, according to the `fadedTokenSight` option (see below for options documentation), sight can be disabled or changed to a different type for faded tokens.

When tokens are faded, their normal settings for opacity, sight type, and the has sight checkbox are saved in a string list in the following format: `opacity, sight_type, has_sight`

This is stored by default on the token in the `libRefData.elevation.fadeData` property. This property can be changed in the `elevation.getTokenFadePropName` method.

Options

There is a simple options API using `elevation.getMapOption` and `elevation.setMapOption`. This is a simple JSON object on the map's data token with keys (option names) and the option value, where a missing key is treated as '0'. Here are the available options:

Option	Setting	Description
fadedSightType	0	Faded tokens (tokens on other elevations) will not have their sight settings changed in any way.
	1	'Has Sight' will be unchecked for all faded tokens.
	STRING	A string matching a defined Sight Type will be applied to faded tokens.
heightFactor	N	This is the height multiplier the new elevation dialog uses to recommend a new elevation value. That dialog sets a fallback value of '2' (so for a 5 ft square, it will recommend new layers at 10 ft), but this value might also be used to determine how a future token height implementation might determine when to display a token below the current layer.
useFogHack	0	Fog of war is not affected by elevation changes.
	1	Fog of war is totally restored on elevation changes, followed by immediately exposing all player token sight ranges. Use with 'fadedSight = 1' to give a more horror-like feel to multi-story buildings. If draw/eraseFoW functions are added, this may be retired.

Sample `libRefData.elevation.layers` object

A trivial example object may look like this:

```
{
 "A14EF2FE619B42AF828CBBAC0192131A": {
    "0": {
        "label": "Ground level",
        "vbl": "JSON Shapes Array",
        "mbl": "JSON Shapes Array",
        "tokens": {
            "BC752679AF784C7DB19EA9ACCF05A362": "JSON Token Appearance Object",
            "19EA9ACCF05A362BC752679AF784C7DB": "JSON Token Appearance Object"
        }
    },
    "60": {
        "label": "Cliff top",
        "vbl": "JSON Shapes Array",
        "mbl": "JSON Shapes Array",
        "tokens": {
            "BC752679AF784C7DB19EA9ACCF05A362": "JSON Token Appearance Object",
            "19EA9ACCF05A362BC752679AF784C7DB": "JSON Token Appearance Object"
         }
     }
 },
 "28CBBAC0192131AA14EF2FE619B42AF8": {
    "0": {
        "label": "Sea Level",
        "vbl": "JSON Shapes Array",
        "mbl": "JSON Shapes Array"
    },
    "10": {
        "label": "Flying",
        "vbl": "JSON Shapes Array",
         "mbl": "JSON Shapes Array"
    }
```

```
  }
 }
```

Retrieved from "https://wiki.rptools.info/index.php?title=Lib:Elevation_Manual&oldid=11494"
