Understanding 3DXChat World Files

By Amber Marlow - Digital Architect & World Builder

Welcome to my comprehensive guide to understanding 3DXChat world files. This document describes how worlds are structured, how to manipulate them, and how to understand the syntax of world files. When you finish reading this, you'll know exactly how a world file works, how to edit it, and what to avoid breaking.

1. What Is a .world File?

A .world file is the complete digital blueprint of a 3DXChat world. It contains every object, coordinate, color, rotation, and material used in the build. There's no hidden code or external assets involved. Everything is right there in structured JSON format. Opening a .world file in any text editor will reveal the "building plan" for your world.

Each .world file is generated by the 3DXChat World Editor and captures a perfect snapshot of the world at that moment in time.

2. Root Structure (What am I really looking at though?)

At the highest level, the .world file defines global environment settings and then lists every object in the scene. Example below:

```
{
  "respawn": { "p": [x,y,z], "r": deg },
  "ambient": [11 floats],
  "oceanlevel": float,
  "weather": "PresetName",
  "valuetype": "float",
  "objects": [ ObjectNode, ... ]
}
```

Key Fields Explanation:

respawn - Determines where the player loads into the world. p is position [x,y,z], and r is the rotation angle in degrees.

ambient - This 11-float array directly corresponds to the sky and lighting presets in the 3DXChat world editor. Each value represents a brightness level for a specific skybox or environmental preset, such as clear skies, sunrise, sunset, overcast, night, rain, or space. By adjusting these values, you control the overall brightness and intensity of each sky type. The engine uses these values to balance the scene's exposure and mood depending on which preset is active. Lower values produce darker, moodier scenes, while higher values brighten everything globally. This array essentially defines how bright or dim your world appears under each available skybox.

oceanlevel - Sets the height of the global water plane. Adjust this to raise or lower water in your world.

weather - A lighting preset like Space, Pink, or Sunny. Each one affects the skybox and illumination.

valuetype - Always "float". It tells the engine how to interpret numeric data.

objects - The collection of everything in your world. This is where all geometry, props, and groups are defined.

3. Object Nodes

Every object, from a chair to a light or a mountain, is defined as an Object Node.

```
ObjectNode := {
  "n": string,          // object name (must exist in-game)
  "p": [x,y,z],         // position
  "r": [rx,ry,rz],      // rotation in degrees
  "s": [sx,sy,sz],      // scale
  "c": [r,g,b],         // color (0–1)
  "m": string,          // material
  "objects": [ ... ]    // children if this is a group
}
```

Important Details

n (Name): The object's name must exactly match one of 3DXChat's internal object names. You can't invent new ones. If the name doesn't exist in the game, that object won't appear in the world, or the world may simply fail to load. There are no warnings or errors, the editor will just ignore the invalid entry.

p (Position): Defines the object's coordinates in 3D space. The Y-axis is vertical.

r (Rotation): Defines the object's orientation in degrees. Most objects require rotation data, even if it's [0,0,0].

s (Scale): Defines object size. Many objects can be scaled, but some, especially built-in decorative or complex game models cannot. The smallest visible scale the game supports is 0.001. Anything smaller (like 0.0001) will automatically revert to 0.001 when the world is saved and reloaded. The editor simply won't render or maintain anything thinner than that. However, when moving objects using snap-step, movements can occur at 0.0001 increments. Below that, movement behaves as if snapping is off entirely.

In practice, you don't have control over decimal precision—the numbers you see don't always reflect what you visually experience. Visual alignment and aesthetics matter far more than mathematical precision. If it looks right, it is right.

c (Color): Color tint applied to the object. Values are normalized between 0 and 1.

m (Material): The surface type used for rendering like GlassRefrThick, MetalOldCopper, or Illum FLAT.

objects: Used only when the object is a group containing other nodes.

4. The Coordinate System

The game uses a simple coordinate system. Y is up. X and Z form the ground plane.

Rotations are expressed in Euler angles: [x, y, z] in degrees. Yaw (the Y-axis) increases clockwise when viewed from above.

5. Environment & Weather

The weather setting changes the global environment preset, which controls the lighting and skybox. Options like Space, Night, or Sunny dramatically alter the atmosphere.

oceanlevel controls the sea height. Set it negative for dry land or higher for submerged builds.

6. Efficiency and File Size (When scripting or creating tools to manipulate .world files)

World files can get large quickly, especially with thousands of primitives. A few smart habits can help:

- Use groups to organize related objects, like buildings or props.
- Avoid unnecessary precision. Six decimal places are plenty.
- Minify files when exporting or scripting them to remove whitespace.

Note that while trimming empty data helps, most objects will still require full `p`, `r`, and `s` fields. A few rare objects, especially decorative assets that can't be rotated or scaled, you may omit these safely. For nearly everything else, they're required.

7. Validation Before Import

Always validate your file before loading it into the game. A missing comma or invalid object name can cause the world to fail to load. There won't be any warning or error message, the editor will just skip loading the world entirely.

Checklist: - Valid JSON formatting - All required root keys are present (`respawn`, `ambient`, `oceanlevel`, `weather`, `valuetype`, `objects`) - Every node includes a valid `n` name from the game's internal list - Colors are between 0 and 1 - Scale values are positive and not zero - Respawn Y coordinate is above the ground plane.

8. Hardcoded Object Names

3DXChat uses a fixed list of available objects built into the game. These include primitives (`Box`, `Cylinder`, `Heart`, `Egg`, etc.) and themed props (`Grass1`, `Rock2`).

If you try to reference an object name that isn't in the game, it will either fail to load or be ignored. When editing `.world` files manually, always double-check that each object name exactly matches the in-game preset.

9. Example Snippets

Empty Group

```
{"n":"group","objects":[]}
```

Simple Object

```
{"n":"Box","p":[0,1,0],"r":[0,0,0],"s":[1,1,1],"c":[0.1,0.1,0.1],"m":"MetalOldCopper"}
```

Grouped Structure

```
{"n":"group","objects":[
  {"n":"Box","p":[0,1,0],"s":[2,1,2],"c":[0.12,0.12,0.12],"m":"GlassRefrThick"},
  {"n":"Cylinder","p":[0,2.5,0],"r":[90,0,0],"s":[0.1,0.1,2],"c":[1,0.9,0.2],"m":"Illum FLAT"}
]}
```

10. Final Word

If you made it this far and still have questions, you now have all the tools to find the answers.
Thank you for taking the time to read this guide on understanding .world files. If you feel overwhelmed by all of this, then just take a deep breath and know that understanding this stuff won't make you a better builder by any margin of measurement. This guide is icing on a cake, without it, it's still a good cake. =)

Much love,
Amber Marlow