Macroverse Truffle Box¶

This box comes with everything a developer needs to useMacroverse in an Ethereum dapp or game.

What is Macroverse¶

Macroverse is a procedurally generated universe, available to be used by blockchain-based games. Rather than paying expensive gas costs to store environment and level data on the blockchain, or spending large amounts of developer effort on custom world generation algorithms, game developers can use Macroverse as a piece of game development middleware, and can find settings for their games by exploring the single, shared Macroverse universe. Macroverse provides trusted, proven world generation algorithms, and a shared universe accessible to all developers, allowing even small developers to make big games.

Access to the Macroverse system is controlled by a token,MRV , which functions as a software license. An Ethereum account must hold a minimum balance, currently100 MRV , in order to query the Macroverse system on the live Ethereum network.

Macroverse Game Ideas¶

- Develop an on-chain version ofKonquest
- , played in a different galactic sector every time.
- Build a galaxy-spanning space trading game, with commodities in each station tradeable using smart contracts.
- Create a single-player 4X game, played in the browser. Take payments from players in Ether, with no app store needed.
- Use procedurally-generated terrain from Macroverse planets as the setting for an off-road racing simulator.
- Devise an orbital-mechanics-based strategy game, set around the moons of a gas giant. Make gameplay proceed in real time, and use blockchain technology to stop cheaters.

Macroverse Development Status¶

Currently, only the first major release of Macroverse, code-namedCannon , is deployed on the live Ethereum network. This release includes a galaxy of over 200 billion procedural stars, but doesnot currently implement planetary systems, orbital mechanics, or planetary terrain. Development of the next release,Kepler , can be tracked in theMacroverse Github repository . Bugs found in the Macroverse contracts should also be reported there.

Getting Started¶

Use this box with:

truffle unbox NovakDistributed/macroverse-truffle-box This will create a default Macroverse-enabled Truffle project, with themacroverse NPM module installed as a dependency, and with a Truffle migration script to enable testnet deployment of the Macroverse Generator and Macroverse Registry contracts.

Macroverse Star Generator Technical Background¶

The Macroverse Star Generator is a smart contract, deployed at address0xc9650c155Bb268A1667B5F9c68701638cAE93d3f , which is responsible for the procedural generation of the galaxy of stars that comprise the Macroverse universe. The galaxy is divided into sectors measuring 25 lightyears on a side. The universe as a whole extends 10,000 sectors in both directioons in X, Y, and Z from the origin, but most of the stars are concentrated in the galaxy, which consists of a disk with a radius of 6,800 sectors in the XZ plane, and a central spherical bulge with a radius of 1000 sectors.

From the X, Y, and Z coordinates (in integers) of each sector, the number of objects (stars, black holes, or similar) in the sector is generated. From the sector coordinates and the object index, the object's seed is generated. The seed of an object serves as its unique identifier for the virtual real estate system, and is also used to generate its properties.

Each object in the Macroverse world is assigned anObjectClass from the following list:

- Supergiant
- Giant
- MainSequence
- WhiteDwarf
- NeutronStar
- BlackHole

Actual stars are assigned aSpectralType according to theHarvard Spectral Classification system.

Additionally, each object has an X, Y, Z position within its sector in light years, and a mass in solar masses. These are fractional values; since Solidity does not provide native support for non-integer types, these values are represented as fixed-point values (stored in a Solidityint128) with 40 fractional bits. TheRealMath Solidity library is provided for working with these values in smart contracts, and themacroverse NPM module provides utility functions for converting to and from them in JavaScript code.

Finally, each object has a flag indicating whether there are planets orbiting it. The actual planetary systems cannot yet be generated (they are coming in theKepler phase of Macroverse development), but the planetary system generation logic, when deployed, will respect this flag.

Querying the Macroverse Star Generator from JavaScript¶

// Make sure you have the Macroverse JavaScript library const mv = require("macroverse")

// Get the Macroverse contract // In Truffle testing code (assuming you have Truffle'sartifacts in scope), you would do: // const MacroverseStarGenerator = artifacts.require("MacroverseStarGenerator")

// In real code, using truffle-contract const TruffleContract = require("truffle-contract") const MacroverseStarGenerator = TruffleContract(require("macroverse/build/contracts/MacroverseStarGenerator.json"))

// You may need to point the contract at your web3 provider here

// Either way, you need to get the deployed instance of the contract const generator = await MacroverseStarGenerator.deployed()

// Then you can get the number of objects in a sector let objectCount = generator.getSectorObjectCount.call(0, 1, -1)

for (let i = 0; i < objectCount; i++) { // Then you can loop over the objects and get the seed for each one let seed = await generator.getSectorObjectSeed.call(0, 1, -1, i)

// And then you can get properties of the object, like its class let objectClass = await generator.getObjectClass.call(seed)

// Use the Macroverse JS module to convert integers -> object class names console.log("Object " + i + " is a " + mv.objectClasses[objectClass]) } The full Macroverse Star Generator API is documented inline in thesmart contract source .