HODGETTS Thomas A

[Company name]  [Company address]

File io Documentation

# Game Objects

## Serializing

Each game object is automatically serialized when a save scene function is called but this will only contain limited data that is shared by all game objects:

* Position
* Rotation
* Scale
* Identifier
* Cool engine ID
* Game Object Type

To save any other data custom serializing functions will have to be added. To enable custom serializing on a game object, the first step is to add a serialize function. One can be overridden from the parent of each game object adding the following line to a header file of choice.



void Serialize(nlohmann::json& jsonData);

Next create the body of the function and save each variable you wish to save into the json file. The first variables you can save are any contained in the class the game object inherits from. This means any variables that are not unique to this type of game object are saved for you. This can be done like below:

Text

Description automatically generated

For any data that is unique to the game object type you are working with can and will have be saved manually. This can be done by taking the type you wish to save and adding a section in the json file and copying the data into that section. Sections do not have to specified beforehand, if they do not exist, the json library will create them for you. The json library supports: strings, integers, floats, doubles, uint64\_t and bools. An example of a string and int type is given below.

Diagram

Description automatically generated

Diagram

Description automatically generated

It also supports arrays of each type so if you wanted a position save a position you can copy the position into an array and repeat the method for copying a variable into the json.



If you wish to input a vector into the json, you can use a for loop to copy the new data into the json. Using the push back function built into the vector class, this will produce a array within the json file that can be read back later.

Graphical user interface, website

Description automatically generated

A full example function should look like this below:

Graphical user interface, text

Description automatically generated

## Deserialize

Each game object is automatically deserialized when a load scene function is called but this will only contain limited data that is shared by all game objects:

* Position
* Rotation
* Scale
* Identifier
* Cool engine ID
* Game Object Type

To any other data custom data a deserializing constructors will have to be added. To enable custom deserializing on a game object, the first step is to add the deserializing constructor. This constructor takes a reference to a json file, and a cool engine ID.



GameObject(const nlohmann::json& data, CoolUUID uuid);

To read any custom data out of a json, all a developer must do is specify the section and read out the data. This will then allow for the data to be assigned to a variable. If it is an array all a user must do is specify a position in that array. Examples are provided below:

Graphical user interface, text

Description automatically generated

Text

Description automatically generated with medium confidence