objects for use with matching in game data objects, whereby the data in the attributes of each serialised data object is copied to a corresponding attribute of the game object, for which the serialised data object is intended. In this manner, attributes of programmed in game data objects may be modified by external data in spreadsheets.

[025] Advantageously, all code generation and the output of one or more serialised data objects is automatic, without requiring code to be written by the game developer, thereby reducing the workload to manually create code that can export external data stored in spreadsheet documents during the game development stage

[026] The automatic code generator to output serialised data objects is described in greater detail below in conjunction with Figures 1 to 8.

[027] Figure 1 shows a workflow 100 to import external data into a software game engine, such as but not limited to Unity®. Steps 102 and 104 are directed at the design of a spreadsheet that is used to contain the external data. Steps 106, 108, 110, 112 and 114 are performed by a platform hosting an automatic code generator, in accordance with an embodiment of the invention, to create serialised data objects from the external data contained in the spreadsheet from the steps 102 and 104. Steps 116 and 118 are performed by the software game engine, which imports the serialised data objects created in the step 114.

[028] In the steps 102 and 104, a methodology is adopted where the data structure of a spreadsheet is defined and the external data is arranged to follow the data structure. This methodology, which for example predefines how the external data is to be organised into columns of specific data type, standardises the creation of spreadsheets that will allow the generated code of the platform to work with the spreadsheet.

[029] Figure 2A shows a spreadsheet 200 (shown as two halves due to space constraints) configured in accordance with one possible implementation of this methodology. The spreadsheet 200 has a table 202 comprising the external data organised in cells 206. The spreadsheet 200 also contains an index 204 accompanying the table 202.

[030] The index 204 defines the data stored in each of the cells 206 in the table 202 through stating one or more properties 208 of the stored data. The index 204 is referred to determine which of the one or more properties 208 impact the code generation that occurs in the steps 106, 108, 110, 112 to create serialised data objects. The one or more properties specify one or more of, but not limited to: a field name for each of the cells 206, the type of data stored in the cell 206, and whether the data in the cell 206 is for use on a client or a server. For instance, all cells under the column “Example_Float” has four properties: a specification (“ID”) of a unique identifier of the serialised data object to which the data in the cell belongs; a specification (“Example_Float”) of a name for the data stored in the cell which also provides the name (see 508 in Figure 5, 608 in Figure 6 and 708 in Figure 7) of an attribute of the serialised data object to which the data in the cell is assigned; a specification (“float”) of the type of data stored in the cell, being in this case float data type; and a specification (“both”) of the scope of

the data stored in the cell, which in this case indicates it is for use in a server or a client. Similarly, all cells under the column "Example_Vector2" has five properties: a specification ("ID") of a unique identifier of the serialised data object to which the data in the cell belongs; a specification ("Example_Vector2") of a name for the data stored in the cell which also provides the name (see 510 in Figure 5, 610 in Figure 6) of an attribute of the serialised data object to which the data in the cell is assigned; a specification ("Vector2") of the type of data stored in the cell, being in this a two dimensional array; a specification ("client") of the scope of the data stored in the cell, which in this case indicates it is for use in only a client; and a specification ("IsNullOrEmpty") of whether a nullity check is to be performed to determine whether there is a requirement for the cells to store data, which in this case is performed for the cells that fall under the column "Example_Vector2".

[031] The external data in the table 202 is considered validly arranged when it is organised in accordance with the structure defined by the index 204. For the spreadsheet 200 of Figure 2A, the structure is as follows. The first row specifies a field name for the data in the cells under its column. The second row specifies the type of data stored in the cells under its column. The third row specifies the scope of the data stored in the cells under its column, because some field names are only used by the client, while others are only used by the server for memory optimization. The fourth row provides a means to specify a test condition that can be imposed to check whether the data stored in the cells under its column is validly configured. The test condition illustrated in spreadsheet 200 is a nullity check ("IsNullOrEmpty"). However, other test conditions (not shown) may be specified, such as, but not limited to: whether there is a data type match in the cell 206 against the data type that the cell 206 is defined to store (e.g. if the second row specifies that the data type as being an integer, verifying that the data in the cell 206 is indeed of integer type and not a float type or string type) and whether there is conformity between the data in one or one more of the cells 206 that are interrelated, applicable when these one or more interrelated cells are required to define an attribute of a game object (e.g. if the data in one cell 206 indicates that the gender of a player character is a female, then test for whether the data in one or more interrelated cells 206 also contain data that supports the indicated gender, such as a female name).

[032] It will be appreciated that other structures are possible, depending on the content of the index 204. Figure 2B shows another spreadsheet 250 (shown as two halves due to space constraints). Like the spreadsheet 250 of Figure 2A, the spreadsheet 250 has a table 252 comprising the external data organised in cells 256. The spreadsheet 250 also contains an index 254 accompanying the table 252. The index 254 defines the data stored in each of the cells 256 in the table 252 through stating one or more properties 258 of the stored data. The index 254 thus defines the data structure of the spreadsheet 250, whereby the external data is arranged to follow the data structure during configuration of the spreadsheet 250.

CLAIMS

1. A platform for creating serialised data objects from external data, the platform configured to generate code that

analyses a table, comprising the external data organised in cells, to identify type of data stored in each cell, wherein the data type is identified through reference to an index accompanying the table, the index defining the data stored in each of the cells in the table through stating one or more properties of the stored data, the table and the index being contained together in an external file;

determines if the identified data types are parsable;

generates, in response to the identified data types being parsable, further code comprising executable statements catered for each of the identified data types that when compiled outputs at least one serialised data object, wherein the data stored in each cell of the table is assigned to an attribute of the serialised data object; and

retrieves an export format for the serialised data object,

wherein the platform is further configured to compile the generated further code to output the serialised data object in the retrieved export format.

2. The platform of claim 1, wherein the index is analysed when producing the further code to create the serialised data object to determine a combination of different data types for use as the attributes for the serialised data object.

3. The platform of claim 1 or 2, wherein the one or more properties specify one or more of: a field name for each of the cells, the type of data stored in the cell, and whether the data in the cell is for use on a client or a server.

4. The platform of any one of the claims 1 to 3, wherein the generated code is configured to differentiate between the index providing data definitions and the table comprising the external data.

5. The platform of any one of the claims 1 to 4, wherein analysis of the table further comprises determining validity of data in each cell against its respective one or more properties.

6. The platform of any one of the preceding claims, wherein analysis of the table further comprises performing a nullity check for cells required to store data and generate an error message in response to detection of blankness.

7. The platform of any one of the preceding claims, wherein analysis of the table further comprises performing a check for duplicate field names.

8. The platform of any one of the preceding claims, further configured to generate an error message in response to the identified data types being determined to be imparsable.

9. The platform of any one of the preceding claims, wherein the data types that the generated code is designed to parse comprise basic data types and custom data types used by a software game engine.

10. The platform of claim 9, wherein the basic data types include any one or more of integer, string, float and Boolean; and the custom data types include any one or more of two and three dimensional vectors and are script specific.

11. The platform of any one or more of the preceding claims, wherein a name for each of the attributes for the serialised data object is obtained from a field name for the cell providing the data for that attribute.

12. The platform of any one or more of the preceding claims, wherein the attributes include a unique identifier for the serialised data object.

13. The platform of any one or more of the preceding claims, further configured to process a cell in the table containing multiple data sets to assign each of them to an attribute of the serialised data object.

14. The platform of any one of the preceding claims, further configured to locate the table for analysis from a plurality of tables through receipt of a search string comprising the name of the table.

15. The platform of any one or more of the preceding claims, further configured to adapt the output serialised data object for use in either a server or a client.

16. The platform of any one of the preceding claims, wherein the export format is one of JSON, scriptable object, CSV.

17. The platform of any one of the preceding claims, wherein the export format is retrieved from a configuration file.

18. A method for creating serialised data objects from external data, the method comprising the steps of:

generating code to analyse a table, comprising the external data organised in cells, to identify type of data stored in each cell to determine if the identified data types are parsable, wherein the data type is identified through reference to an index accompanying the table, the index defining the data stored in each of the cells in the table through stating one or more properties of the stored data, the table and the index being contained together in an external file;

generating, in response to the identified data types being parsable, further code comprising executable statements catered for each of the identified data types that when compiled outputs at least one serialised data object, wherein the data stored in each cell of the table is assigned to an attribute of the serialised data object;

retrieving an export format for the serialised data object; and

compiling the generated further code to output the serialised data object in the retrieved export format.

19. The method of claim 18, further comprising;

organising the external data into the cells in accordance with an index accompanying the table, the index defining the data stored in each of the cells in the table through stating one or more properties of the stored data.