Package: mvdf (via r-universe)

December 18, 2024

Type Package

Title A Minimum Viable Data Format for 3D Rendering via Blender

Version 0.0.0.9000

Description A small, self-contained, minimum viable data format providing a standard interface for using R as a front-end for the Blender 3D rendering program. The core approach centers around an S4 class, 'mvdf', with getter, setter, and validation methods designed to be extended for more specific rendering approaches.

URL https://mikemahoney218.github.io/mvdf/

BugReports https://github.com/mikemahoney218/mvdf/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Config/testthat/edition 3

VignetteBuilder knitr

Suggests knitr, codemetar, pkgdown, rmarkdown, lintr, styler, testthat, dplyr, brio, devtools

SystemRequirements Blender

Depends R (>= 3.5)

Imports methods, rlang, glue, proceduralnames

Collate 'add_blender_endmatter.R' 'add_camera.R' 'mvdf_class.R' 'add_empty.R' 'add_light.R' 'mvdf_simple_material.R' 'add_primitive.R' 'add_render_image.R' 'appendix.R' 'assign.R' 'base_methods.R' 'create_blender_frontmatter.R' 'execute_render.R' 'get_mvdf.R' 'metadata.R' 'mvdf-package.R' 'show.R' 'zzz.R'

Config/pak/sysreqs blender

Repository https://mikemahoney218.r-universe.dev

RemoteUrl https://github.com/mikemahoney218/mvdf

RemoteRef HEAD

RemoteSha 22a299354846376a1d0013771d6c26090a112899

Contents

`add_blender_endmatter` *Add standard ending boilerplate to a Blender rendering script*

Description

Add standard ending boilerplate to a Blender rendering script

Usage

```
add_blender_endmatter(script, filepath = tempfile(fileext = ".blend"))
```

Arguments

`script`	The Python script to append the generated code onto.
`filepath`	The file path to save the output file to. Must end with '.blend'.

Value

A length 1 character vector containing the Blender Python script with ending boilerplate added.

`add_camera` *Add a camera object to a Blender scene*

Description

Add a camera object to a Blender scene

Usage

```
add_camera(
  script,
  align = c("WORLD", "VIEW", "CURSOR"),
  location = c(0, 0, 0),
  rotation = c(0, 0, 0),
  convert_rotations = TRUE
)
```

Arguments

`script`	The Python script to append the generated code onto.
`align`	Character: alignment of the new camera object. Options include "WORLD" (align the new object to the world), "VIEW" (align the new object to the view), and "CURSOR" (use the 3D cursor orientation for the new object).
`location`	A numeric vector with length 3 specifying the x, y, and z grid coordinates for the new camera.
`rotation`	A numeric vector with length 3 specifying the x, y, and z rotation for the new camera.
`convert_rotations`	Logical: convert 'rotation' to radians? Set to 'FALSE' if 'rotation' is already in radians.

Value

A length 1 character vector containing the Blender Python script with code to add a camera object added.

add_empty *Add code to create empties to a Blender script*

Description

This function generates code that will, when run inside Blender, create empties within a scene. See the official Blender documentation at https://docs.blender.org/api/blender_python_api_current/bpy.ops.object.html for a full list of available options.

Usage

```
add_empty(script, object, type = "PLAIN_AXES", location = NULL, ...)

add_empty_method(object, script, type = "PLAIN_AXES", location = NULL, ...)

## S4 method for signature 'mvdf_obj'
add_empty_method(object, script, type = "PLAIN_AXES", location = NULL, ...)
```

Arguments

script	The Python script to append the generated code onto.
object	An object inheriting from [mvdf_obj] which will be used to calculate the x, y, and z positions for each primitive. One primitive will be created for each row of ‘mvdf(object)‘.
type	The empty type to create. Options include 'PLAIN_AXES', 'ARROWS', 'SINGLE_ARROW', 'CIRCLE', 'CUBE', 'SPHERE', 'CONE', and 'IMAGE'.
location	Either ‘NULL‘ (the default) or a vector of strings (in the format ‘location=(x, y, z)‘) specifying the location for the origin of each empty. If ‘NULL‘, this vector will be automatically calculated using the ‘x‘, ‘y‘, and ‘z‘ values in ‘object‘.
...	Additional arguments to pass to the empty creation call.

Value

A length 1 character vector containing the Blender Python script with code for creating mesh primitives added.

add_light *Add a light object to a Blender scene*

Description

Add a light object to a Blender scene

Usage

```
add_light(
  script,
  type = c("POINT", "SUN", "SPOT", "AREA"),
  radius = 0,
  align = c("WORLD", "VIEW", "CURSOR"),
  location = c(0, 0, 0),
  rotation = c(0, 0, 0),
  convert_rotations = TRUE,
  energy = 10
)
```

Arguments

script	The Python script to append the generated code onto.
type	Character: Type of light source to use. Options include "POINT" (omnidirectional point light source), "SUN" (constant direction parallel ray light source), "SPOT" (directional cone light source), and "AREA" (directional area light source).
radius	Numeric: Radius of the light source.
align	Character: alignment of the new light object. Options include "WORLD" (align the new object to the world), "VIEW" (align the new object to the view), and "CURSOR" (use the 3D cursor orientation for the new object).
location	A numeric vector with length 3 specifying the x, y, and z grid coordinates for the new light.
rotation	A numeric vector with length 3 specifying the x, y, and z rotation for the new light.
convert_rotations	Logical: convert ‘rotation‘ to radians? Set to ‘FALSE‘ if ‘rotation‘ is already in radians.
energy	Numeric: The power ("wattage") of the light object.

Value

A length 1 character vector containing the Blender Python script with code to add a light object added.

---

add_primitive *Add code to add primitives to a Blender script*

Description

This function generates code that will, when run inside Blender, create mesh primitives within a scene.

Usage

```
add_mesh_primitive(
  script,
  object,
  primitive = "ico_sphere",
  location = NULL,
  ...
)

add_surface_primitive(
  script,
  object,
  primitive = "torus",
  location = NULL,
  ...
)

add_curve_primitive(
  script,
  object,
  primitive = "bezier_circle",
  location = NULL,
  ...
)
```

Arguments

`script`	The Python script to append the generated code onto.
`object`	An object inheriting from [mvdf_obj] which will be used to calculate the x, y, and z positions for each primitive. One primitive will be created for each row of ‘mvdf(object)‘.
`primitive`	The primitive type to create. See the official Blender documentation at https://docs.blender.org/api/blender_python_api_current/bpy.ops.html for a full list of available primitives. Each category has a different set of primitives available.
`location`	Either ‘NULL‘ (the default) or a vector of strings (in the format ‘location=(x, y, z)‘) specifying the location for the origin of each primitive. If ‘NULL‘, this vector will be automatically calculated using the ‘x‘, ‘y‘, and ‘z‘ values in ‘object‘.
`...`	Additional arguments to pass to the primitive creation call. The available arguments are different for each primitive, and are documented in the official Blender documentation at https://docs.blender.org/api/blender_python_api_current/bpy.ops.mesh.html.

Value

A length 1 character vector containing the Blender Python script with code for creating primitives added.

add_render_image *Add steps to render a scene to an image file to a Blender script*

Description

This function will add the necessary steps to render a Blender scene and save the rendering to an image file. Note that this process requires you have a camera and a light source in your scene!

Usage

```
add_render_image(script, filepath, ...)
```

Arguments

script	The Python script to append the generated code onto.
filepath	The file path to save the output file to. Note that the file format to use is inferred from the file extension; using non-standard or abbreviated extensions might result in render errors. Allowed formats include BMP, IRIS, PNG, JPEG, JPEG2000, TARGA, TARGA_RAW, CINEON, DPX, OPEN_EXR_MULTILAYER, OPEN_EXR, HDR, TIFF, AVI_JPEG, AVI_RAW, FRAMESERVER, H264, FFMPEG, THEORA, XVID.
...	Any additional options to pass to ‘bpy.ops.render.render‘.

Value

A length 1 character vector containing the Blender Python script with render-to-image code added.

appendix *Retrieve the appendix from a ‘forthetrees‘ object*

Description

Retrieve the appendix from a ‘forthetrees‘ object

Usage

```
appendix(object)

## S4 method for signature 'mvdf_obj'
appendix(object)
```

Arguments

object	The object to retrieve the appendix from.

`appendix<-` *Set appendix values for an 'mvdf_obj' object.*

Description

Set appendix values for an 'mvdf_obj' object.

Usage

```
appendix(x) <- value

## S4 replacement method for signature 'mvdf_obj'
appendix(x) <- value
```

Arguments

`x`	The object to set the appendix within.
`value`	The data to replace the appendix with.

`as.data.frame,mvdf_obj-method` *Coerce to a data frame*

Description

Coerce any object inheriting from 'mvdf' to a data frame.

Usage

```
## S4 method for signature 'mvdf_obj'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	An object inheriting from 'mvdf'.
`row.names, optional, ...`	Arguments passed to [base::as.data.frame].

create_blender_frontmatter *Create a Blender rendering script with standard beginning boilerplate*

Description

Create a Blender rendering script with standard beginning boilerplate

Usage

```
create_blender_frontmatter(
  imports = c("bpy", "mathutils", "math"),
  delete = c("Cube", "Camera", "Light"),
  before = NULL,
  after = NULL
)
```

Arguments

imports	A character vector of Python packages to import at the start of the script. Packages are imported without aliases, so that script-building functions may always expect to use the full package name. Defaults to 'bpy', 'mathutils', and 'math'. Set to 'NULL' to not import any packages.
delete	A character vector of objects in the default Blender scene to delete. Defaults to all objects in the default scene ('Cube', 'Camera', and 'Light'). Set to 'NULL' to not delete any objects. Note that not deleting the initial cube may result in your file failing to save due to a known bug in Blender.
before	Optionally, a character vector containing lines of code to add before package imports.
after	Optionally, a character vector containing lines of code to add after object deletion.

Value

A length 1 character vector containing the introductory boilerplate for a Blender Python script.

execute_render *Execute a Python script in Blender*

Description

This function takes a Python script and executes it via the Blender 3D rendering program. While the intent of this function is to take scripts generated by 'mvdf' and related packages and use them to produce 3D renderings, neither this function nor Blender validates the script – so be careful when executing Python scripts you haven't generated yourself, as scripts run in Blender have the same filesystem permissions as scripts you run on the terminal!

Usage

```
execute_render(script, blender = NULL, flags = NULL, addons = NULL)
```

Arguments

script	The Python script (either as a file path or as a character vector with length 1) to execute.
blender	Path to the Blender executable to execute the Python script. If 'NULL', the default, uses the first result from 'Sys.which("blender")'.
flags	Additional command-line arguments to pass to 'blender'.
addons	A vector of Blender add-ons to enable on the command line.

Value

A length 1 character vector with the output file path is returned invisibly if the function can identify the file the Python script saves to. If the output file can't be identified, returns 'NULL', invisibly.

---

head,mvdf_obj-method *Return the first or last parts of objects inheriting from 'mvdf_obj'*

Description

Return the first or last parts of objects inheriting from 'mvdf_obj'

Usage

```
## S4 method for signature 'mvdf_obj'
head(x, n = 6L, ...)

## S4 method for signature 'mvdf_obj'
tail(x, n = 6L, ...)
```

Arguments

x	An object inheriting from 'mvdf_obj'
n	an integer vector of length up to nrow(mvdf(x)). Values specify the indices to be selected along the rows of the object. A positive value of n[i] includes the first/last n[i] indices in that dimension, while a negative value excludes the last/first abs(n[i]), including all remaining indices. NA or non-specified values (when length(n) < length(dim(x))) select all indices in that dimension. Must contain at least one non-missing value.
...	Ignored.

metadata *Retrieve the metadata data frame from a 'forthetrees' object*

Description

Retrieve the metadata data frame from a 'forthetrees' object

Usage

```
metadata(object)

## S4 method for signature 'mvdf_obj'
metadata(object)
```

Arguments

object	The object to retrieve the metadata data frame from.

metadata<- *Set metadata values for an 'mvdf_obj' object.*

Description

Set metadata values for an 'mvdf_obj' object.

Usage

```
metadata(x) <- value

## S4 replacement method for signature 'mvdf_obj'
metadata(x) <- value
```

Arguments

x	The object to set the metadata within.
value	The data to replace the metadata with.

mvdf *Retrieve mvdf values from a ‘mvdf_obj‘ object.*

Description

Retrieve mvdf values from a ‘mvdf_obj‘ object.

Usage

```
mvdf(object)

## S4 method for signature 'mvdf_obj'
mvdf(object)
```

Arguments

object	The ‘mvdf_obj‘ object to retrieve the mvdf for.

mvdf<- *Set mvdf values for an ‘mvdf_obj‘ object.*

Description

Set mvdf values for an ‘mvdf_obj‘ object.

Usage

```
mvdf(x) <- value

## S4 replacement method for signature 'mvdf_obj'
mvdf(x) <- value
```

Arguments

x	The object to set the mvdf within.
value	The data to replace the mvdf with.

mvdf_obj *Construct a Minimum Viable Data Frame object*

Description

Construct a Minimum Viable Data Frame object

Usage

```
mvdf_obj(
  data = NULL,
  x = "x",
  y = "y",
  z = "z",
  idx = "idx",
  metadata = NULL,
  appendix = NULL
)
```

Arguments

data	Optionally, a data frame containing all the data necessary to create a ‘mvdf_obj‘. If left ‘NULL‘, then ‘x‘, ‘y‘, ‘z‘, and ‘idx‘ are interpreted as the values to use for each slot; if not ‘NULL‘, ‘x‘, ‘y‘, ‘z‘, and ‘idx‘ are interpreted as the names of columns in ‘data‘ containing the values for each slot.
x, y, z	Numeric: distance of the origin of the object from the origin of the grid system (the central point at 0, 0, 0) in meters in the given direction. Must have no ‘NA‘, ‘NULL‘, ‘NaN‘, ‘Inf‘, or ‘-Inf‘ values. If ‘data‘ is not ‘NULL‘, the names of columns in ‘data‘ with values for the respective slot. Coordinates are assumed to be on a right-handed coordinate system with Z oriented as the natural "vertical" direction.
idx	Character: a unique identifier (or "index") for each object to be modeled. Must be unique with no ‘NA‘ or ‘NULL‘ values, but otherwise is not validated. If ‘data‘ is not ‘NULL‘, the names of columns in ‘data‘ with values for the slot. If left ‘NULL‘, a sequential index is generated.
metadata	Data frame: a table containing additional information on the objects to be modeled. Optional, but if this slot is used then the data frame must contain a column named ‘idx‘ which should correspond to the ‘idx‘ slot. Only the existence of this column is validated.
appendix	List: additional data produced in the generation of the object. Not validated; any additional outputs that don’t map to modeled objects may be inserted here.

mvdf_obj-class *The Minimum Viable Data Frame S4 class*

Description

The Minimum Viable Data Frame S4 class

Slots

`x,y,z` Numeric: distance of the origin of the object from the origin of the grid system (the central point at 0, 0, 0) in meters in the given direction. Must have no ‘NA‘, ‘NULL‘, ‘NaN‘, ‘Inf‘, or ‘-Inf‘ values. Coordinates are assumed to be on a right-handed coordinate system with Z oriented as the natural "vertical" direction.

`idx` Character: a unique identifier (or "index") for each object to be modeled. Must be unique with no ‘NA‘ or ‘NULL‘ values, but otherwise is not validated.

`metadata` Data frame: a table containing additional information on the objects to be modeled. Optional, but if this slot is used then the data frame must contain a column named ‘idx‘ which should correspond to the ‘idx‘ slot. Only the existence of this column is validated.

`appendix` List: additional data produced in the generation of the object. Not validated; any additional outputs that don’t map to modeled objects may be inserted here.

See Also

Other classes and related functions: `mvdf_simple_material-class`

mvdf_simple_material *Create a ‘mvdf_simple_material‘ object*

Description

Create a ‘mvdf_simple_material‘ object

Usage

```
mvdf_simple_material(
  data = NULL,
  diffuse_color = "diffuse_color",
  metallic = "metallic",
  roughness = "roughness",
  translate_colors = NULL,
  ...
)
```

Arguments

`data`	Optionally, a data frame containing all the data necessary to create a ‘mvdf_simple_material‘. If ‘NULL‘, all other arguments are interpreted as data to use in constructing the object; if not ‘NULL‘, arguments are interpreted as the names of columns in ‘data‘ containing the values for each slot.
`diffuse_color`	Diffuse color of the material, in either a RGBA array (if ‘translate_colors‘ is ‘TRUE‘) or in any of the formats understood by [grDevices::col2rgb] (if ‘translate_colors‘ is ‘FALSE‘). If colors are missing, they are set to gray80. If ‘translate_colors‘ is ‘NULL‘, the default, this function attempts to infer if values are already RGBA arrays.
`metallic`	Amount of mirror reflection for raytrace, as a float from 0-1. If missing, set to 0.
`roughness`	Roughness of the material, as a float from 0-1. If missing, set to 0.
`translate_colors`	Logical: use ‘grDevices‘ to create RGBA arrays from ‘diffuse_color‘?
`...`	Additional arguments passed to [mvdf_obj]

`mvdf_simple_material-class` *Class to attach basic material data to ‘mvdf_obj‘ objects*

Description

Class to attach basic material data to ‘mvdf_obj‘ objects

Slots

`x,y,z` Numeric: distance of the origin of the object from the origin of the grid system (the central point at 0, 0, 0) in meters in the given direction. Must have no ‘NA‘, ‘NULL‘, ‘NaN‘, ‘Inf‘, or ‘-Inf‘ values. Coordinates are assumed to be on a right-handed coordinate system with Z oriented as the natural "vertical" direction.

`idx` Character: a unique identifier (or "index") for each object to be modeled. Must be unique with no ‘NA‘ or ‘NULL‘ values, but otherwise is not validated.

`metadata` Data frame: a table containing additional information on the objects to be modeled. Optional, but if this slot is used then the data frame must contain a column named ‘idx‘ which should correspond to the ‘idx‘ slot. Only the existence of this column is validated.

`appendix` List: additional data produced in the generation of the object. Not validated; any additional outputs that don’t map to modeled objects may be inserted here.

`diffuse_color` Diffuse color of the material, as an RGBA array of floats scaled from 0-1.

`metallic` Amount of mirror reflection for raytrace, as a float from 0-1.

`roughness` Roughness of the material, as a float from 0-1.

See Also

Other classes and related functions: `mvdf_obj-class`

nrow,mvdf_obj-method *The Number of Rows/Columns of an mvdf*

Description

'nrow' and 'ncol' return the number of rows or columns present in 'mvdf(x)'.

Usage

```
## S4 method for signature 'mvdf_obj'
nrow(x)

## S4 method for signature 'mvdf_obj'
ncol(x)
```

Arguments

x	An object inheriting from 'mvdf_obj'

Value

An integer of length 1 or NULL.

plot,mvdf_obj-method *Generic X-Y Plotting*

Description

Draw a basic paired scatter plot from the mvdf of an object inheriting from 'mvdf_obj'.

Usage

```
## S4 method for signature 'mvdf_obj'
plot(x, y, ...)
```

Arguments

x	Any object inheriting from mvdf_obj.
y, ...	Arguments passed to base 'plot'.

setvalues *Set values for objects subclassing 'mvdf_obj'*

Description

This function returns a new object of the same class as 'object' with updated values. Use it as a convenient, pipe-able way to set values for objects subclassing 'mvdf_obj'.

Usage

```
set_values(
  object,
  mvdf = NULL,
  metadata = NULL,
  appendix = NULL,
  newclass = NULL,
  ...
)

set_mvdf(mvdf, object, metadata = NULL, appendix = NULL, newclass = NULL, ...)

set_metadata(
  metadata,
  object,
  mvdf = NULL,
  appendix = NULL,
  newclass = NULL,
  ...
)

set_appendix(
  appendix,
  object,
  mvdf = NULL,
  metadata = NULL,
  newclass = NULL,
  ...
)
```

Arguments

object	The object to update.
mvdf	The minimum viable data frame required by the S4 class. If 'NULL' (the default), uses the mvdf from 'object'.
metadata	The metadata to include in the new object. If 'NULL' (the default), uses the metadata from 'object'.

appendix	The appendix to include in the new object. If 'NULL' (the default), uses the appendix from 'object'.
newclass	The class of the object to return. If 'NULL' (the default), returns an object of class 'class(object)'.
...	Any additional arguments used in the constructor function being called.

Value

An S4 object (of class 'newclass' if specified or 'class(object)' if not) with updated values.

`show,mvdf_obj-method` *Show an object inheriting from 'mvdf_obj'*

Description

Display the object, by printing, plotting or whatever suits its class.

Usage

```
## S4 method for signature 'mvdf_obj'
show(object)
```

Arguments

object	Any object inheriting from 'mvdf_obj'

Value

'show' returns an invisible 'NULL'.

Index