Celaria Map Toolkit

Iceflower S

Jan 13, 2022

CONTENTS

CHAPTER

ONE

USER GUIDE

1.1 Installation

1.1.1 System Requirements

Python is required in the version 3.7. or higher, it can be downloaded at https://www.python.org/downloads/.

During the Windows installation you should make sure that the `PATH` / environment variable is set and `pip` is installed.

Under Linux it should be ensured that `pip` is installed, if this is not done with the standard installation.

1.1.2 Installation

```
pip install cmt
```

1.2 Using cmt

The program is a terminal program, so it runs from the terminal.

Calling with: .. code-block:: none

cmt

-h, --help
show this help message and exit

-v, --version
show program's version number and exit

convert `file {cmap,ecmap} {0,1} output`
convert file to type, version and output file

CHAPTER

TWO

INTERNALS

Reference material.

2.1 cmt

2.1.1 cmt.a_converter

class cmt.a_converter.**AConverter**

Bases: abc.ABC

abstract static convert(*source*)

Convert to the other map format of same version.

Return type *AMap*

abstract static downgrade(*source*)

Downgrade to the format version below.

Return type *AMap*

abstract static upgrade(*source*)

Upgrade to the format version above.

Return type *AMap*

2.1.2 cmt.a_map

class cmt.a_map.**AMap**(*identifier*, *version*)

Bases: abc.ABC

abstract classmethod decode(*data*, *offset*, *debug=False*)

Return type *AMap*

abstract encode()

Return type bytearray

class cmt.a_map.**MapType**(*value*)

Bases: enum.Enum

An enumeration.

CMAP = 'celaria_map'

ECMAP = 'celaria_edi'

static from_str(*text*)

Return type *MapType*

2.1.3 cmt.convert

cmt.convert.**convert**(*source*, *version*, *target*)
First convert to the target type and down/upgrade to the correct version. :raises ValueError: something failed

Return type *AMap*

2.1.4 cmt.decode

raises ValueError something failed

rtype *AMap*

2.1.5 cmt.encode

2.1.6 cmt.static_data

2.1.7 cmt.utils

class cmt.utils.**DebugIterUnpack**(*format_*, *buffer*, *what*)
Bases: object

cmt.utils.**debug_print**(*data*, *what*, *value*, *offset=None*)

cmt.utils.**to_hex**(*data*)

cmt.utils.**unpack_from**(*format_*, *buffer*, *offset*, *what*, *debug*)
Same behaviour as struct.unpack_from.

Parameters

- **format** –
- **buffer** (bytes) –
- **offset** (int) –
- **what** (Tuple[str, ...]) – tuple of message for every unpacked value
- **debug** (bool) – use debug mode

Returns

2.2 cmt.blender

2.2.1 cmt.blender.v2_80

cmt.blender.v2_80.import_menu

cmt.blender.v2_80.object_panel

cmt.blender.v2_80.scene_panel

cmt.blender.v2_80.utils

cmt.blender.v2_80.viewport_add_menu

2.2.2 cmt.blender.v2_80.v1

cmt.blender.v2_80.v1.add_objects

cmt.blender.v2_80.v1.codec

2.3 cmt.cmap

2.3.1 cmt.cmap.a_cmap

class cmt.cmap.a_cmap.**ACMap**(*version*)

Bases: *cmt.a_map.AMap*

abstract classmethod decode(*data*, *offset*, *debug=False*)

Return type *ACMap*

abstract encode()

Return type bytearray

2.3.2 cmt.cmap.a_entity

class cmt.cmap.a_entity.**AEntity**(*type_*, *byte_size*)

Bases: abc.ABC

Variables

- **type** – entity type
- **byte_size** – size in bytes the entity uses

abstract classmethod decode(*data*, *offset*, *debug=False*)

Parameters

- **data** (bytes) –

- **offset** (int) – without entity type byte
- **debug** (bool) –

Return type *AEntity*

abstract encode()
Includes the entity type.

Return type bytearray

2.4 cmt.cmap.v0

2.4.1 cmt.cmap.v0.cmap

class cmt.cmap.v0.cmap.**CMap**
Bases: *cmt.cmap.a_cmap.ACMap*

Celaria .cmap format (version 0)

Datatypes

Abbreviation	Type	Byte size
uByte	unsigned byte	1
uShort	unsigned short	2
uInt	unsigned int	4
sShort	signed short	2
sInt	signed int	4
f32	float	4
f64	double	8

Description format

```
> <datatype> (<number of datatypes in sequence>) // <description>
```

or

```
> [<variable name>] :  <datatype> (<number of datatypes in sequence>) //
<description>
```

Format

```
> uByte (11) // string identifier
> uByte (1) // version

> nameLen : uByte (1) // number of characters in map name
> uByte (nameLen) // map name as String

> uByte (1) - boolean, if the timer will be run in singleplayer

> uByte (1) // unused byte

> times : uByte (1) - number of checkpoint times (including medal time)

> uInt (times) // checkpoint times for platin
```

(continues on next page)

(continued from previous page)

```
> uInt (times) // checkpoint times for gold
> uInt (times) // checkpoint times for silver
> uInt (times) // checkpoint times for bronze

> f32 (1) // sun rotation on Z axis
> f32 (1) // sun height expressed as an angle (between 0 and 90 degrees)

> f64 (1) // preview camera position x
> f64 (1) // preview camera position y
> f64 (1) // preview camera position z
> f64 (1) // preview camera look at position x
> f64 (1) // preview camera look at position y
> f64 (1) // preview camera look at position z

> entityNumber : uInt (1) // number of entities on the map

for entity in entityNumber {
    > entityType : uInt (1) // entityType

    switch(entityType) {
        case 0: // block
            > blockType : uByte (1) // blockType/color
            > uByte (1) // unused byte
            > sInt (1) // position x
            > sInt (1) // position y
            > uInt (1) // position z
            > uInt (1) // scale x
            > uInt (1) // scale y
            > uInt (1) // scale z
            > f32 (1) // rotation on Z axis

            if (blockType == 5){ // checkpoint block
                > uByte (1) // checkpoint Number
            }

        case 1: // sphere
            > sInt (1) // position x
            > sInt (1) // position y
            > sInt (1) // position z

        case 2: // player start
            > uByte (1) // unused byte
            > sInt (1) // position x
            > sInt (1) // position y
            > sInt (1) // position z
            > f32 (1) // rotation on Z axis

        case 128: // dummy id
            > uByte (1) // ID
            > sInt (1) // position x
            > sInt (1) // position y
            > uInt (1) // position z
```

(continues on next page)

(continued from previous page)

```
            > uInt (1) // scale x
            > uInt (1) // scale y
            > uInt (1) // scale z
            > f32 (1) // rotation on Z axis
}
```

classmethod decode(*data*, *offset*, *debug=False*)

Return type *CMap*

encode()

Return type bytearray

2.4.2 cmt.cmap.v0.entities

class cmt.cmap.v0.entities.**Block**

Bases: *cmt.cmap.a_entity.AEntity*

classmethod decode(*data*, *offset*, *debug=False*)

Parameters

- **data** (bytes) –
- **offset** (int) – without entity type byte
- **debug** (bool) –

Return type *Block*

encode()

Includes the entity type.

Return type bytearray

class cmt.cmap.v0.entities.**BlockType**(*value*)

Bases: enum.Enum

An enumeration.

CHECKPOINT = 5

FINISH = 1

ICE = 4

JUMP = 2

NOTHING = 0

SPEED = 3

class cmt.cmap.v0.entities.**Dummy**

Bases: *cmt.cmap.a_entity.AEntity*

classmethod decode(*data*, *offset*, *debug=False*)

Parameters

- **data** (bytes) –
- **offset** (int) – without entity type byte
- **debug** (bool) –

Return type *Dummy*

encode()
Includes the entity type.

Return type bytearray

class cmt.cmap.v0.entities.**PlayerStart**
Bases: *cmt.cmap.a_entity.AEntity*

classmethod decode(*data*, *offset*, *debug=False*)

Parameters

- **data** (bytes) –
- **offset** (int) – without entity type byte
- **debug** (bool) –

Return type *PlayerStart*

encode()
Includes the entity type.

Return type bytearray

class cmt.cmap.v0.entities.**Sphere**
Bases: *cmt.cmap.a_entity.AEntity*

classmethod decode(*data*, *offset*, *debug=False*)

Parameters

- **data** (bytes) –
- **offset** (int) – without entity type byte
- **debug** (bool) –

Return type *Sphere*

encode()
Includes the entity type.

Return type bytearray

2.4.3 cmt.cmap.v0.medal_time

class cmt.cmap.v0.medal_time.**MedalTime**(*platin=0*, *gold=0*, *silver=0*, *bronze=0*)
Bases: object

cmt.cmap.v0.medal_time.**decode_medal_times**(*data*, *offset*, *debug=False*)
Must start with the length byte.

Parameters

- **data** (bytes) –
- **offset** (int) –
- **debug** (bool) –

Return type List[*MedalTime*]

2.5 cmt.cmap.v1

2.5.1 cmt.cmap.v1.cmap

class cmt.cmap.v1.cmap.**CMap**
Bases: *cmt.cmap.a_cmap.ACMap*

Celaria .cmap format (version 1)

Datatypes

Abbreviation	Type	Byte size
uByte	unsigned byte	1
uShort	unsigned short	2
uInt	unsigned int	4
sShort	signed short	2
sInt	signed int	4
f32	float	4
f64	double	8

Description format

```
> <datatype> (<number of datatypes in sequence>) // <description>
```

or

```
> [<variable name>] :  <datatype> (<number of datatypes in sequence>) //
<description>
```

Format

```
> uByte (11) // string identifier
> uByte (1) // version

> nameLen : uByte (1) // number of characters in map name
> uByte (nameLen) // map name as String

> uByte (1) // unused - gamemode
```

(continues on next page)

(continued from previous page)

```
> uByte (1) // number of checkpoint times (including finish line)

> times : uByte (1) // number of checkpoint times (including finish line)

> uInt (times) // checkpoint times for platin
> uInt (times) // checkpoint times for gold
> uInt (times) // checkpoint times for silver
> uInt (times) // checkpoint times for bronze

> f32 (1) // sun rotation on Z axis
> f32 (1) // sun angle to xy plane (between 0 and 90 degrees)

> f64 (1) // preview camera position x
> f64 (1) // preview camera position y
> f64 (1) // preview camera position z
> f64 (1) // preview camera look at position x
> f64 (1) // preview camera look at position y
> f64 (1) // preview camera look at position z

> entityNumber : uInt (1) // number of entities on the map

for entity in entityNumber {
    > entityType : uInt (1) // entityType

    switch(entityType) {
        case 0: // block
            > blockType : uByte (1) // blockType/color
            > sInt (1) // position x
            > sInt (1) // position y
            > uInt (1) // position z
            > uInt (1) // scale x
            > uInt (1) // scale y
            > uInt (1) // scale z
            > f32 (1) // rotation on Z axis

            if (blockType == 5){ // checkpoint block
                > uByte (1) // checkpoint Number
            }

        case 1: // sphere
            > sInt (1) // position x
            > sInt (1) // position y
            > uInt (1) // position z

        case 2: // player start
            > uByte (1) // unknown
            > sInt (1) // position x
            > sInt (1) // position y
            > uInt (1) // position z
            > f32 (1) // rotation on Z axis
```

(continues on next page)

(continued from previous page)

```
        case 128: // dummy id
            > uByte (1) // ID
            > sInt (1) // position x
            > sInt (1) // position y
            > uInt (1) // position z
            > uInt (1) // scale x
            > uInt (1) // scale y
            > uInt (1) // scale z
            > f32 (1) // rotation on Z axis
    }
}
```

classmethod decode(*data*, *offset*, *debug=False*)

Return type *CMap*

encode()

Return type bytearray

2.5.2 cmt.cmap.v1.entities

class cmt.cmap.v1.entities.**Block**

Bases: *cmt.cmap.a_entity.AEntity*

classmethod decode(*data*, *offset*, *debug=False*)

Parameters

- **data** (bytes) –
- **offset** (int) – without entity type byte
- **debug** (bool) –

Return type *Block*

encode()

Includes the entity type.

Return type bytearray

class cmt.cmap.v1.entities.**BlockType**(*value*)

Bases: enum.Enum

An enumeration.

CHECKPOINT = 5

FINISH = 1

ICE = 4

JUMP = 2

NOTHING = 0

SPEED = 3

class cmt.cmap.v1.entities.**Dummy**
Bases: *cmt.cmap.a_entity.AEntity*

classmethod decode(*data*, *offset*, *debug=False*)

Parameters

- **data** (bytes) –
- **offset** (int) – without entity type byte
- **debug** (bool) –

Return type *Dummy*

encode()
Includes the entity type.

Return type bytearray

class cmt.cmap.v1.entities.**PlayerStart**
Bases: *cmt.cmap.a_entity.AEntity*

classmethod decode(*data*, *offset*, *debug=False*)

Parameters

- **data** (bytes) –
- **offset** (int) – without entity type byte
- **debug** (bool) –

Return type *PlayerStart*

encode()
Includes the entity type.

Return type bytearray

class cmt.cmap.v1.entities.**Sphere**
Bases: *cmt.cmap.a_entity.AEntity*

classmethod decode(*data*, *offset*, *debug=False*)

Parameters

- **data** (bytes) –
- **offset** (int) – without entity type byte
- **debug** (bool) –

encode()
Includes the entity type.

Return type bytearray

2.5.3 cmt.cmap.v1.checkpoint_time

class cmt.cmap.v1.checkpoint_time.**CheckpointTime**(*platin=0*, *gold=0*, *silver=0*, *bronze=0*)
Bases: object

cmt.cmap.v1.checkpoint_time.**decode_checkpoint_times**(*data*, *offset*, *debug=False*)
Must start with the length byte.

Parameters

- **data** (bytes) –
- **offset** (int) –
- **debug** (bool) –

Return type List[*CheckpointTime*]

2.6 cmt.converter

2.6.1 cmt.converter.v0

2.6.2 cmt.converter.v1

2.7 cmt.cs

2.7.1 cmt.cs.main

cmt.cs.main.**main**(*argv=None*)
Entry point into the program. Gets the arguments from the console and proceed them with ArgumentParser. Returns if its success successful 0 else 1.

2.8 cmt.ecmap

2.8.1 cmt.ecmap.a_ecmap

class cmt.ecmap.a_ecmap.**AECMap**(*version*)
Bases: *cmt.a_map.AMap*

abstract classmethod decode(*data*, *offset*, *debug=False*)

Return type *AECMap*

abstract encode()

Return type bytearray

2.9 cmt.ecmap.v0

2.9.1 cmt.ecmap.v0.ecmap

class cmt.ecmap.v0.ecmap.**ECMap**

Bases: *cmt.ecmap.a_ecmap.AECMap*

Celaria .ecmap format (version 0)

Datatypes

Abbreviation	Type	Byte size
uByte	unsigned byte	1
uShort	unsigned short	2
uInt	unsigned int	4
sShort	signed short	2
sInt	signed int	4
f32	float	4
f64	double	8

Description format

```
> <datatype> (<number of datatypes in sequence>) // <description>
```

or

```
> [<variable name>] :  <datatype> (<number of datatypes in sequence>) //
<description>
```

Difference regarding to the .cmap begins with a '!!'.

Comparing to CMap v0.

Format

```
> uByte (11) // string identifier
> uByte (1) // version

> nameLen : uByte (1) // number of characters in map name
> uByte (nameLen) // map name as String

!! // checkpoint times are missing

> f32 (1) // sun rotation on Z axis
> f32 (1) // sun angle to xy plane (between 0 and 90 degrees)

... same as cmap v0 ...
```

classmethod decode(*data*, *offset*, *debug=False*)

Return type ECMAP

encode()

Return type bytearray

2.10 cmt.ecmap.v1

2.10.1 cmt.ecmap.v1.ecmap

class cmt.ecmap.v1.ecmap.**ECMap**

Bases: *cmt.ecmap.a_ecmap.AECMap*

Celaria .ecmap format (version 1)

Datatypes

Abbreviation	Type	Byte size
uByte	unsigned byte	1
uShort	unsigned short	2
uInt	unsigned int	4
sShort	signed short	2
sInt	signed int	4
f32	float	4
f64	double	8

Description format

```
> <datatype> (<number of datatypes in sequence>) // <description>
```

or

```
> [<variable name>] :  <datatype> (<number of datatypes in sequence>) //
<description>
```

Difference regarding to the .cmap begins with a '!!'.

Comparing to CMap v1.

Format

```
> uByte (11) // string identifier
> uByte (1) // version

> nameLen : uByte (1) // number of characters in map name
> uByte (nameLen) // map name as String

> uByte (1) // unused - gamemode

!! // checkpoint times are missing

> f32 (1) // sun rotation on Z axis
> f32 (1) // sun angle to xy plane (between 0 and 90 degrees)

... same as cmap v1 ...
```

classmethod decode(*data*, *offset*, *debug=False*)

Return type ECMAP

encode()

Return type bytearray

2.11 cmt.ecmap.v2

2.11.1 cmt.ecmap.v2.ecmap

class cmt.ecmap.v2.ecmap.**ECMap**

Bases: *cmt.ecmap.a_ecmap.AECMap*

Celaria .ecmap format (version 2)

Datatypes

Abbreviation	Type	Byte size
uByte	unsigned byte	1
uShort	unsigned short	2
uInt	unsigned int	4
sShort	signed short	2
sInt	signed int	4
f32	float	4
f64	double	8

Description format

```
> <datatype> (<number of datatypes in sequence>) // <description>
```

or

```
> [<variable name>] :  <datatype> (<number of datatypes in sequence>) //
<description>
```

Difference regarding to the .cmap begins with a '!!'.

Comparing to CMap v1.

Format

```
> uByte (11) // string identifier
> uByte (1) // version

> nameLen : uByte (1) // number of characters in map name
> uByte (nameLen) // map name as String

!! > uByte (1) // boolean, previewCam_set

> uByte (1) // unused - gamemode

!! // checkpoint times are missing

> f32 (1) // sun rotation on Z axis
> f32 (1) // sun angle to xy plane (between 0 and 90 degrees)

... same as cmap v1 ...
```

classmethod decode(*data*, *offset*, *debug=False*)

Return type ECMAP

encode()

Return type bytearray

- genindex
- modindex

PYTHON MODULE INDEX

C

INDEX

Symbols

A

B

C

D

E

F

I

J

M

N

P

S

T

U