CoreEngine API

Version 1.0

Rohde Nils

core-control.de

Content

[1 Changelog 2](#_Toc390630292)

[2 Licenses 2](#_Toc390630293)

[2.1 CoreEngine 2](#_Toc390630294)

[2.1 Other used products 2](#_Toc390630295)

[4 API 3](#_Toc390630296)

[4.1 CCoreDialog 3](#_Toc390630297)

[4.1.1 Method summary 3](#_Toc390630298)

[4.5 ObjectTemplate 3](#_Toc390630299)

[4.6 CCoreObjectHelper 4](#_Toc390630300)

[4.6.1 Method summary 4](#_Toc390630301)

# 1 Changelog

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date of change | Author | Improvements |
| 1.0 | 15.06.2014 | Nils Rohde | Creating of this document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 2 Licenses

## 2.1 CoreEngine

The CoreEngine (in the following as “engine”) is distributed under the Beerware license for non commercial use. If you wish to use the engine in a commercial context, please feel free to ask for it with an email to [nils.rohde@core-control.de](mailto:nils.rohde@core-control.de) . The Beerware license is attached to this document.

## 2.1 Other used products

# 4 API

## 4.1 CCoreDialog

This class can generate neat user input dialogs. Later there should also be a method for creating simple output only dialogs. Every method is static, so you don’t need the engine running to generate a dialog.

### 4.1.1 Method summary

string CCoreDialog::promt(string windowCaption, string message)

This method generates a user input dialog. The input from the user is returned as std::string. The method can be called without a loaded engine. It takes a std::string for the window title and one for the printed message. The method is locking, so you cant do anything within the main thread if the prompt is open. It can be closed either by clicking on the button or by hitting return.

## 4.5 ObjectTemplate

This structure is for storing object information. This information is set by the user and stored in the /media/objects.cof. Normally you don’t have to mess with this, because every usage is handled by the CCoreObjectHelper class

struct ObjectTemplate

{

string name;

string normalGfx;

string hoverGgfx;

string objectType;

};

# 4.6 CCoreObjectHelper

In order to use CCoreobjects you have to initialize a CCoreObjectHelper by calling the standard constructor. The Objecthelper stores and manages onscreen objects.

### 4.6.1 Method summary

CCoreObjectHelper();

Constructor, initializes the Objecthelper

bool loadFromFile();

Loads all available objects from /media/objects.cof. Is also called by the constructor, but can be called to refresh the list of available object templates. Returns false if the file could not be opened or another error occurs. Returns also false if there were no entrys in the template list.

int createObject(string name);

Creates the real screen object and stores it. It takes the name of the desired object (same as defined in /media/objects.cof). If the object template was not found it returns 9999. Else it returns the unique object ID, which you should store for further usage.

bool deleteObject(int index);

Deletes the with index specified screen objet (Note: index is the unique objectID, which the createObject method returns). It returns false if the object was not found or an error occurred.