Simulation 模擬

A simulation in CoppeliaSim can be started, paused and stopped with *[Menu bar --> Simulation --> Start/Pause/Stop simulation] or through the related toolbar buttons:*

可以使用[菜單欄->模擬->開始/暫停/停止模擬]或通過相關的工具欄按鈕來啟動，暫停和停止 CoppeliaSim 中的模擬：

[Simulation start/pause/stop toolbar buttons]

Internally, the simulator will use additional intermediate states in order to correctly inform scripts or programs about what will happen next. Following state diagram illustrates the simulator's internal states:

[模擬開始/暫停/停止工具欄按鈕]

在內部，模擬器將使用其他中間狀態，以正確告知腳本或程序接下來將發生的情況。以下狀態圖說明了模擬器的內部狀態：

Scripts and programs should alwaysreact according to the current system call function and possibly the simulation state in order to behave correctly. It is good practice to divide each control code into at least 4 system call functions (e.g. for non-threaded child scripts):

腳本和程序應始終根據當前系統調用功能以及可能的模擬狀態進行反應，以便正確運行。優良作法是將每個控制代碼至少分為 4 個系統調用函數（例如，用於非線程子腳本）：

Initialization function sysCall_init: the function is called only when the script is initialized.

- **Actuation function sysCall_actuation**: the function is called when actuation should happen.
- **Sensing function sysCall_sensing**: this function is called when sensing should happen.
- **Clean-up function sysCall_cleanup**: the function is called just before the script is de-initialized (e.g. at simulation end, or when the script is destroyed).

・初始化函數：sysCall_init：僅在腳本初始化時才調用該函數。

・激活函數：sysCall_actuation：應在發生激活時調用該函數。

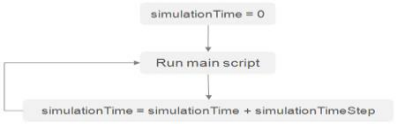
・感應功能：sysCall_sensing：應在感應發生時調用此功能。

・清理函數：sysCall_cleanup：該函數在腳本未初始化之前被調用（例如，在仿真結束時或銷毀腳本時）。

For examples on how to arrange a typical script, refer to the main script, the child scripts and customization scripts pages.

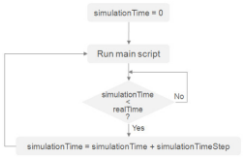
有關如何安排典型腳本的示例，請參閱主腳本，子腳本和自定義腳本頁面。SIMULATION LOOPThe simulator operates by advancing the simulation time at constant time steps. Following figure illustrates the main simulation loop:

模擬器通過以恆定的時間步長推進模擬時間來進行操作。下圖說明了主要的仿真循環：



Real-time simulation is supported by trying to keep the simulation time synchronized with the real time:

通過嘗試使仿真時間與實時保持同步來支持實時仿真：



Following represents a very simplified main client application (messaging, plugin handling and other details have been omitted for clarity purpose):

以下是一個非常簡化的主客戶端應用程序（為清晰起見，已省略了消息，插件處理和其他詳細信息）：

程式碼 PORGRAM

```
void initializationCallback

{

    // do some initialization here

}



void loopCallback



    if ( (simGetSimulationState()&sim simulation advancing)!=0 )

    {

        if ( (simGetRealTimeSimulation()!=1)||(simIsRealTimeSimulationStepNeeded()==1) )

        {

            if ((simHandleMainScript()&sim script main script not called)==0)

                simAdvanceSimulationByOneStep();

        }

    }

}
```

```
void deinitializationCallback

{

   // do some clean-up here

}
```

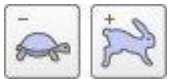
Depending on the simulation complexity, performance of the computer and simulation settings, real-time simulation might not always be possible.

取決於模擬的複雜性，計算機的性能和模擬設置，實時模擬可能並不總是可能的。

Simulation speed

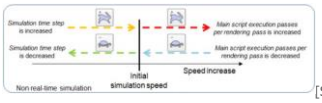
In non real-time simulations, the simulation speed (i.e. the perceived speed) is mainly dependent on two factors: the simulation time step and the number of simulation passes for one rendering pass (see the simulation dialog for more details). In the case of a real-time simulation, the simulation speed mainly depends on the real-time multiplication coefficient, but also to a certain degree of the simulation time step (a too small simulation time step might not be compatible with the real-time character of a simulation because of the limited calculation power of the computer). During simulation, the simulation speed can be adjusted with following toolbar buttons:

在非實時仿真中，仿真速度（即感知速度）主要取決於兩個因素：仿真時間步長和一個渲染通道的仿真通道數量（有關更多詳細信息，請參見仿真對話框）。在實時仿真的情況下，仿真速度主要取決於實時乘法係數，而且在一定程度上取決於仿真時間步長（太小的仿真時間步長可能與實時時間不兼容）。由於計算機的計算能力有限，因此無法進行仿真。在模擬過程中，可以使用以下工具欄按鈕來調整模擬速度：



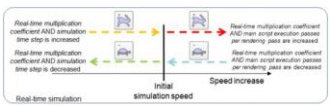
[Simulation speed adjustment toolbar buttons]The simulation speed is adjusted in a way so that the initial simulation time step is never increased (because this might have as consequence the breaking of a mechanism for example). Following two figures illustrate the simulation speed adjustment mechanisms:

[模擬速度調整工具欄按鈕] 以某種方式調整模擬速度，以使初始模擬時間步長永遠不會增加（如，這可能因此而導致機制中斷）。以下兩個圖說明了仿真速度調整機制：



[Simulation speed adjustment mechanism for **non real-time simulations**]

[非實時模擬的模擬速度調整機制]



[Simulation speed adjustment mechanism for **real-time simulations**]

By default, each simulation cycle is composed by following **sequential** operations:

[用於實時仿真的仿真速度調整機制] 默認情況下，每個模擬週期由以下順序操作組成：

- Executing themain script
- Rendering the scene・執行主腳本・渲染場景

Threaded rendering

The rendering operation will always increase thesimulation cycle duration, thus also slowing down simulation speed. The number of main script executions per scene rendering can be defined (see further up), but this is not enough in some situations, because rendering will still slow down every xth simulation cycle (which can be handicapping with real-time requirements). For those situations, a threaded rendering mode can be activated via the user settings, or via the following toolbar button:

渲染操作將始終增加仿真週期的持續時間，從而也降低了仿真速度。可以定義每個場景渲染的主腳本執行次數（請參閱後面的內容），但這在某些情況下還不夠，因為渲染仍然會減慢每個第 x 個模擬週期的時間（這可能會限制實時性）。在這種情況下，可以通過用戶設置或以下工具欄按鈕激活線程渲染模式：  [Threaded rendering toolbar button]When the threaded rendering mode is activated, a simulation cycle will only consist in execution of the main script, thus simulations will run at maximum speed. Rendering will happen via a different thread, and not slow down the simulation task. The drawbacks have however to be considered. When threaded rendering is activated, then:

- Rendering will happen asynchronously to the simulation loop, and visual glitches might appear
- Thevideo recorderwill not operate at constant speed (some frames might get skipped)
- The stability of the application might be reduced• Some operations (e.g. erasing an object, etc.) require to wait for the rendering thread to finish work, before being able to execute, and vice-versa. In those situations, cycles could take more time than in the sequential rendering mode[線程渲染工具欄按鈕] 激活線程渲染模式後，模擬週期將僅包含在執行主腳本中，因此模擬將以最大速度運行。渲染將通過不同的線程進行，並且不會減慢模擬任務的速度。然而，必須考慮缺點。激活線程渲染後，： •渲染將與模擬循環異步發生，並且可能會出現視覺故障 •錄像機無法以恆定速度運行（會跳過某些幀） •應用程序的穩定性可能會降低 •某些操作（例如擦除對像等）需要等待渲染線程完成工作才能執行，反之亦然。在那些情況下，循環可能比順序渲染模式花費更多的時

Recommended topics

- Related API functions• Simulation settings dialog• Scripts• The main client application• Plugins•相關的 API 函數 •模擬設置對話框 •腳本 •主要客戶應用 •插件參考於:

https://coppeliarobotics.com/helpFiles/index.html