

RAGE

Tutorials

Task/Timer

Software & Documentation © The RAGE Team 2005

Version Relevance: JGE Release 0.1+

CONTENTS

Are you reading the right tutorial?	3
Required Imports	3
Required Properties	3
Task Orientated Game Loop	4
Using the Timer outside Task	6

ARE YOU READING THE RIGHT TUTORIAL?

This tutorial is intended to show you how to use the task concept model as the basis for your game loop.

The tutorial assumes you have the knowledge of how timers are used to create smooth state transition in games and that you have read the tutorial on how to use properties in the kernel (the timer requires a property be set).

REQUIRED IMPORTS

Task Package: `jge.kernel.task`
Timer Package: `jge.kernel.timer`

REQUIRED PROPERTIES

<i>Property</i>	<i>Description</i>	<i>Example Value</i>
<code>class.Timer</code>	Implementation of <code>jge.kernel.timer.Timer</code> to use for Timers in JGE.	<code>plugin.timer.JavaTimer</code>

TASK ORIENTATED GAME LOOP

The game loop in the Kernel is simulated with a single-thread of tasks that are updated once in the period of a cycle.

Each task has an initialise function, update function and a done function. The initialise function is called on all tasks when they are in the TaskManager and are approaching their first update cycle. The done function is called when the task has been killed (removed from the TaskManager).

The update function is called once each game loop cycle. It is passed a float value representing the amount of seconds that it took to update last time (i.e. the change in time).

HOW TO...

To have code executed in the game loop you need to derive Task and provide implementations for the following functions (returning false on any function will cause the TaskManager to kill all tasks):

```
+init():boolean  
+update(dt:float)boolean  
+done():boolean
```

The code you place in the update function will be executed once each cycle.

If you would like to have different tasks execute before/after other tasks, you can set the priority of a task when calling the Task superclass constructor.

EXAMPLE

In the example my task will print out to the command line "This is My Task running. [dt] secs gone by." The task will run with the highest priority.

```
public MyTask extends Task  
{  
    public MyTask()  
    {  
        super(Task.PRIORITY_MAX);  
    }  
  
    public boolean update(float dt)  
    {  
        System.out.println("This is My Task running. "  
            + dt + " secs gone by.");  
        return true;  
    }  
  
    public boolean init() { return true; }  
    public boolean done() { return true; }  
}
```

USING THE TIMER OUTSIDE TASK

A single timer is used inside the task game loop so that the change in time (dt) can be calculated. Most of the time you will just need to use the time passed to tasks, but if you need to use a timer explicitly, timers are available in the Kernel.

Calling `+getTimer():Timer` on the Kernel will return a Timer implementation for the platform.

All timers have four functions:

- `+getTime():long`
Get the amount of time the timer has been running for, in nanoseconds.
- `+reset():void`
Resets the timer to '0' but doesn't stop it.
- `+start():void`
Starts the timer from its current position (not necessarily '0').
- `+stop():void`
Stops the timer and leaves it at its current position.

EXAMPLE

This code creates and starts a timer, sleeps for 500ms and prints out the timers value.

```
//get the timer from the Kernel
Timer timer = Kernel.getInstance().getTimer();

//start the timer, it is at position '0'
timer.start();

try
{
    Thread.sleep(500);
}
catch(Exception e)
{
    System.out.println("Didn't sleep for 500ms.");
}

// stop the timer so that the next line won't be included
timer.stop();

System.out.println("Slept for: " + timer.getTime()
    + " nanosecs");
```

OTHER UTILITIES

A class `jge.kernel.resource.ResourceInputStream` provides extra functionality for reading `InputStreams`. It wraps the `InputStream` and gives access to pieces of its data in different formats.

You can request from the stream the next int, and it will compile together the byte into an int and return its value. Similar functions exist for other file formats including `Strings`.

It also allows you to read chunks of data using an offset and reading a particular length into an array.

