Time Manager

Now, Hugo can't do actual-real-time games like *Border Zone* or *Knight Orc* where NPCs wander around and things happen whether or not you are entering commands, but it *can* keep track of time. Kent Tessman wrote a number of time-keeping routines for *Future Boy!* (it kept track of how long the player has been playing the game). I've included them in Roodylib. They've been useful for some rather obscure purposes. For instance, the IF interpreter Gargoyle can read configuration files but can't write to them. Saving the current time to file and then reading the time right back (and checking the difference) is a good way to see if an interpreter *fully* supports configuration files. They're also used by Roodylib's "jukebox" to determine if a song has ended.

```
#include "timesystem.h"
```

turning on the time manager

The time system uses a `time_object` class that keeps track of years, months, days, minutes, and so forth.

```
class time_object
{
       tm_year 0
       tm_month 0
       tm_day 0
       tm_hour 0
       tm_minute 0
       tm_second 0
}
```

the `time_object` class

Usually, to do anything, you'll need at least three `time_object` objects.

```
time_object movie_start
{}

time_object current_time
{}

time_object difference_in_time
{}

time_object movie_length
{}
```

example `time_objects`

With the above, you store the time you started a video in `movie_start`, periodically checking the time and storing it in `current_time`, determine the difference between those two and store it in `difference_in_time`, and finally compare *that* to `movie_length` to determine if the movie is over.

Let's go over some time-management routines:

`GetCurrentTime(timefile)`- Saves the current time to the `time_object` given as an argument.

`CalculateTimeDifference(current, previous, result)` - Determines the difference between the current-time `time_object` and the earlier-time `time_object`, saving to result `time_object`.

`IsTimeLonger(first, second)` - Returns true if first `time_object` is longer than second.

`AddTimes(time1, time2, result)` - Adds two `time_objects`, saving to result `time_object`.

`CopyTimeValue(time_orig, time_copy)` - Copies one `time_object` to another.

`PrintTimeValue(time, no_seconds)` - Prints a `time_object` as "# years, # months, # days, # hours, # minutes, # seconds" (if the `no_seconds` argument is true, the seconds are skipped).