CalendarDateAndTimeSystem

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 CalendarController Class Reference

Controls a GUI calander. Contains functions to build and control a calendar.

Inherits MonoBehaviour.

Public Member Functions

- void **YearPrev** ()

  Decrements the displayed year.
- void **YearNext** ()

  Increments the displayed year. </summary
- void **MonthPrev** ()

  Decrements the displayed month.
- void **MonthNext** ()

  Incremenets the displayed month
- bool **is_date_displayed** (DateTime d)

  Returns whether the current date is displayed on the calendar.
- DateTime **get_first_displayed_date** ()

  Get the first displayed date.
- DateTime **get_last_displayed_date** ()

  Get the last displayed date.
- bool **highlight_date** (DateTime date, DAY_STYLE style)
- bool **highlight_date** (DateTime date, Color highlight)

  Highlight DateTime d by setting the highlight color. Highlighted dates can retain their background color.
- bool **highlight_date** (**DateTimeEvent** e, DAY_STYLE style)

  *Highlight **DateTimeEvent*** *(p. 14)* *e by setting the highlight style.*
- void **toggle_visibility** ()

  Alternately toggles the visibility of the calendar window and the minimized icon.
- void **set_visibility** (bool b)

  Sets the visibilty of the calendar to true or false.
- void **OpenWindow** ()

  Opens the calander window
- void **CloseWindow** ()

  Closes the calander window

Public Attributes

- **TimeManager time_manager**

  TimeManager *(p. 17) component attached to this* ***CalendarController*** *(p. 5)*
- Text **date_header**

  The upper text label to display the date and time.
- Image **background_image**

  The background image.
- GameObject **day_prefab**

  The prefab for each day item.
- ColorPalette **pallette**

  Color Palette For This Calendar
- List< **CalendarDateItem** > **date_items** = new List< **CalendarDateItem**>()

  All of the date_items of the currently displayed month

Properties

- DateTime **display_date** `[get, set]`

  The currently displayed date and time.

3.1.1 Detailed Description

Controls a GUI calander. Contains functions to build and control a calendar.

If the property time_manager is set, it will display that date. Otherwise it uses datetime.Now to initilize the displayed date.

Calendars are made up of **CalendarDateItem** (p. 11) objects as well as buttons and other UI elements.

This script automatically updates these elements as time passes.

A working prefab with this script as a component is given in the package.

```
// Highlight a specific day highlight_date(display_date.AddDays(1), DAY_↩
STYLE.Generic);
```

3.1.2 Member Function Documentation

3.1.2.1 CloseWindow()

```
void CalendarController.CloseWindow ( )
```

Closes the calander window

3.1.2.2 get_first_displayed_date()

```
DateTime CalendarController.get_first_displayed_date ( )
```

Get the first displayed date.

Returns

The smallest displayed **CalendarDateItem** (p. 11).

3.1.2.3 get_last_displayed_date()

```
DateTime CalendarController.get_last_displayed_date ( )
```

Get the last displayed date.

Returns

The largest displayed **CalendarDateItem** (p. 11).

3.1.2.4 highlight_date() [1/3]

```
bool CalendarController.highlight_date (
            DateTime date,
            Color highlight )
```

Highlight DateTime d by setting the highlight color. Highlighted dates can retain their background color.

3.1.2.5 highlight_date() [2/3]

```
bool CalendarController.highlight_date (
            DateTime date,
            DAY_STYLE style )
```

Parameters

date	
style	

3.1.2.6 highlight_date() [3/3]

```
bool CalendarController.highlight_date (
              DateTimeEvent e,
             DAY_STYLE style )
```

Highlight **DateTimeEvent** (p. 14) e by setting the highlight style.

Highlighted dates can retain their background color.

3.1.2.7 is_date_displayed()

```
bool CalendarController.is_date_displayed (
             DateTime d )
```

Returns whether the current date is displayed on the calendar.

Parameters

d	The DateTime object to check.

Returns

Returns true if the DateTime object is currently rendered.

3.1.2.8 MonthNext()

```
void CalendarController.MonthNext ( )
```

Incremenets the displayed month

3.1.2.9 MonthPrev()

```
void CalendarController.MonthPrev ( )
```

Decrements the displayed month.

3.1.2.10 OpenWindow()

```
void CalendarController.OpenWindow ( )
```

Opens the calander window

3.1.2.11 set_visibility()

```
void CalendarController.set_visibility (
            bool b )
```

Sets the visibilty of the calendar to true or false.

The 0th child of the calendar controller gameobject should always be the calendar. The 1st child of the calendar controller gameobject should be the minimized icons.SA

Parameters

b	True enables the calendar, false disables the calander and enables the minimized calendar.

3.1.2.12 toggle_visibility()

```
void CalendarController.toggle_visibility ( )
```

Alternately toggles the visibility of the calendar window and the minimized icon.

3.1.2.13 YearPrev()

```
void CalendarController.YearPrev ( )
```

Decrements the displayed year.

3.1.3 Member Data Documentation

3.1.3.1 background_image

```
Image CalendarController.background_image
```

The background image.

3.1.3.2 date_header

```
Text CalendarController.date_header
```

The upper text label to display the date and time.

3.1.3.3 date_items

```
List< CalendarDateItem> CalendarController.date_items = new List< CalendarDateItem>()
```

All of the date_items of the currently displayed month

3.1.3.4 day_prefab

```
GameObject CalendarController.day_prefab
```

The prefab for each day item.

3.1.3.5 pallette

```
ColorPalette CalendarController.pallette
```

Color Palette For This Calendar

3.1.3.6 time_manager

```
TimeManager CalendarController.time_manager
```

TimeManager (p. 17) component attached to this **CalendarController** (p. 5)

3.1.4 Property Documentation

3.1.4.1 display_date

```
DateTime CalendarController.display_date  [get], [set]
```

The currently displayed date and time.

The documentation for this class was generated from the following file:

- CalendarController.cs

3.2 CalendarDateItem Class Reference

CalendarDateItem (p. 11) defines the GUI behaviour of the calendar days.

Inherits MonoBehaviour.

Public Member Functions

- void **configure** (DateTime d)

  Configure the date item for the given datetime.
- void **configure** (DateTime d, DAY_STYLE style)

  *Configure this **CalendarDateItem** (p. 11) with DateTime d and a style.*
- void **set_background_color** (Color c, string tag=null)
- void **set_foreground_color** (Color c, string tag=null)

Public Attributes

- DateTime **date**

  *The date represented by this **CalendarDateItem** (p. 11).*
- Text **day_label**

  Text that displays the current day number
- Image **background**

  *The background image of the **CalendarDateItem** (p. 11).*

Properties

- Color **bg_color** `[get]`

  The background color of the datetime object.
- Color **fg_color** `[get]`

  The foreground color of the datetime object.
- Color **highlight_color** `[get, set]`

  The highlight color of the datetime object.

3.2.1 Detailed Description

CalendarDateItem (p. 11) defines the GUI behaviour of the calendar days.

The color and style of a **CalendarDateItem** (p. 11) can be changed.

The **configure(DateTime d)** (p. 11) function configures the **CalendarDateItem** (p. 11) to represent a certian day.

3.2.2 Member Function Documentation

3.2.2.1 configure() `[1/2]`

```
void CalendarDateItem.configure (
            DateTime d )
```

Configure the date item for the given datetime.

Parameters

d	Datetime to configure the day for.

3.2.2.2 configure() [2/2]

```
void CalendarDateItem.configure (
            DateTime d,
            DAY_STYLE style )
```

Configure this **CalendarDateItem** (p. 11) with DateTime d and a style.

Parameters

d	
style	A DAY_STYLE configures the **CalendarDateItem** (p. 11) to use the **CalendarController** (p. 5) color palette.

3.2.2.3 set_background_color()

```
void CalendarDateItem.set_background_color (
            Color c,
            string tag = null )
```

Parameters

c	

3.2.2.4 set_foreground_color()

```
void CalendarDateItem.set_foreground_color (
            Color c,
            string tag = null )
```

Parameters

c	

3.2.3 Member Data Documentation

3.2.3.1 background

```
Image CalendarDateItem.background
```

The background image of the **CalendarDateItem** (p. 11).

3.2.3.2 date

```
DateTime CalendarDateItem.date
```

The date represented by this **CalendarDateItem** (p. 11).

3.2.3.3 day_label

```
Text CalendarDateItem.day_label
```

Text that displays the current day number

3.2.4 Property Documentation

3.2.4.1 bg_color

```
Color CalendarDateItem.bg_color  [get]
```

The background color of the datetime object.

3.2.4.2 fg_color

```
Color CalendarDateItem.fg_color  [get]
```

The foreground color of the datetime object.

3.2.4.3 highlight_color

```
Color CalendarDateItem.highlight_color  [get], [set]
```

The highlight color of the datetime object.

The highlight color effect mixes the components of the background color and the RGB components of the highlight color.

The alpha component of the highlight color controlls the lerp amount.

The documentation for this class was generated from the following file:

- CalendarDateItem.cs

3.3 DateTimeEvent Class Reference

Inherits ScriptableObject.

Public Member Functions

- void **OnTickEvent** ()

  Invoke the function when the eventTime is reached. The Calendar controller will fire OnTickEvent when the event_time is reached.

Static Public Member Functions

- static **DateTimeEvent** **InstantiateDateTimeEvent** (DateTime event_time, Action method, DAY_STYLE **style**=DAY_STYLE.Generic)

  *Creates a new **DateTimeEvent** (p. 14) and returns it.*

Public Attributes

- Action **function**

  The function to call.
- DAY_STYLE **style**

  The style of the day

Properties

- DateTime **EventTime** [get, set]

  The DateTime that Func<object> function will be called.

3.3.1 Member Function Documentation

3.3.1.1 InstantiateDateTimeEvent()

```
static  DateTimeEvent DateTimeEvent.InstantiateDateTimeEvent (
            DateTime event_time,
            Action method,
            DAY_STYLE style = DAY_STYLE.Generic )  [static]
```

Creates a new **DateTimeEvent** (p. 14) and returns it.

Parameters

date	The firing DateTime of the event.
method	The method to call when the event is ticked.

Returns

An instance of the ticked event.

3.3.1.2 OnTickEvent()

```
void DateTimeEvent.OnTickEvent ( )
```

Invoke the function when the eventTime is reached. The Calendar controller will fire OnTickEvent when the event↩ _time is reached.

3.3.2 Member Data Documentation

3.3.2.1 function

```
Action DateTimeEvent.function
```

The function to call.

3.3.2.2 style

```
DAY_STYLE DateTimeEvent.style
```

The style of the day

3.3.3 Property Documentation

3.3.3.1 EventTime

```
DateTime DateTimeEvent.EventTime  [get], [set]
```

The DateTime that Func<object> function will be called.

The documentation for this class was generated from the following file:

- DateTimeEvent.cs

3.4 Test_Script Class Reference

Inherits MonoBehaviour.

Public Member Functions

- void **lunchtime** (object sender, EventArgs e)

  This function is called at lunchtime every day. Remember to eat lunch!
- void **tgif** (object sender, EventArgs e)

  This function ticks on friday. It exlaims relief on the last day of the work week!

3.4.1 Member Function Documentation

3.4.1.1 lunchtime()

```
void Test_Script.lunchtime (
            object sender,
            EventArgs e )
```

This function is called at lunchtime every day. Remember to eat lunch!

Parameters

sender	The sender object.
e	Event arguments. I have ignored them and passed data through the **TimeManager** (p. 17) MonoBehaviour. You could implement them if you want to pass custom arguments.

3.4.1.2 tgif()

```
void Test_Script.tgif (
            object sender,
            EventArgs e )
```

This function ticks on friday. It exlaims relief on the last day of the work week!

Parameters

sender	The sender object.
e	Event arguments. I have ignored them and passed data through the **TimeManager** (p. 17) MonoBehaviour. You could implement them if you want to pass custom arguments.

The documentation for this class was generated from the following file:

- Test_Script.cs

3.5 TimeManager Class Reference

The **TimeManager** (p. 17) class travels forward in time and fires functions on events.

Inherits MonoBehaviour.

Public Member Functions

- void **AddEventToStack** (**DateTimeEvent** e)

  *Adds the **DateTimeEvent** (p. 14) e to the stack.*
- void **AddEventToStack** (**DateTimeEvent** e, **CalendarController** c, DAY_STYLE d)

  *Adds the **DateTimeEvent** (p. 14) e to the stack and highlights the date on a calendar.*

Public Attributes

- bool **start_now** = false

  *Set to true to intialize the **TimeManager** (p. 17) with DateTime.Now*
- int **start_year** = 1762

  The sim start year. (0 AD $<$ start_year $<$ 9999 AD)
- int **start_month** = 3

  The sim start month (1 $<$ start_month $<$ 12).
- int **start_day** = 14

  The sim start day (1 $<$ start_day $<$ Days in month)
- int **start_hour** = 8

  The sim start hour (0 $<$ start_hour $<$ 23)
- int **start_minute** = 0

  The sim start minute (0 $<$ start_minute $<$ 59)
- float **game_time_multiplier**

  The game time multiplier.

Protected Member Functions

- virtual void **OnTickYear** (DateTime d)

  Event called every year.
- virtual void **OnTickMonth** (DateTime d)

  Event called every month.
- virtual void **OnTickDay** (DateTime d)

  Event called every day.
- virtual void **OnTickNamedDay** (DateTime d)

  Fires events based on what day of the week it is ie. sunday monday tuesday ect.
- virtual void **OnTickQuarterDay** (DateTime d)

  Tick the quarter day events(12am, 6am, 12pm, 6pm)
- virtual void **OnTickHour** (DateTime d)

  Event called every hour.
- virtual void **OnTickQuarterHour** ()

  Event called every 15 miuntes.
- virtual void **OnTickMinute** ()

  Event called every minute. This will be called 1440 times every simulated day.

Properties

- DateTime **start_date** [get]

  The start date of the simulation set by the start_year, start_month, start_day, start_hour and start_minute.
- DateTime **date** [get, set]

  The current date. This value is incremented using Time.DeltaTime, so it is not gaurenteed to be at any specific time.
- SortedSet< **DateTimeEvent** > **event_stack** [get]

  The event stack is a sorted set based on the date. Automatically stores DateTimeEvents in a sorted list based on the date each event should fire..

Events

- EventHandler **tick_minute**
- EventHandler **tick_quarter_hour**
- EventHandler **tick_hour**
- EventHandler **tick_day**
- EventHandler **tick_month**
- EventHandler **tick_year**
- EventHandler **tick_monday**
- EventHandler **tick_tuesday**
- EventHandler **tick_wednesday**
- EventHandler **tick_thursday**
- EventHandler **tick_friday**
- EventHandler **tick_saturday**
- EventHandler **tick_sunday**
- EventHandler **tick_midnight**
- EventHandler **tick_morning**
- EventHandler **tick_noon**
- EventHandler **tick_evening**

3.5.1 Detailed Description

The **TimeManager** (p. 17) class travels forward in time and fires functions on events.

The pass_time function is used to move time forward and call events.

Events can be repeating, like tick_day, which is called every day in simulation time.

There is also an event stack that can call DateTimeEvents.

DateTimeEvents are called once, on their EventDate. They can fire any function when they are called.

DateTimeEvents are sorted into the event stack and evaluated in the **TimeManager** (p. 17) pass_time loop automatically.

How to subscribe to a periodic event: `this.tick_day += print_date; // Event every day this.tick_friday += print_date; // Event every friday this.tick_noon += print_date; // Event every noontime`

`Prints the date to the debug.log console. public void print_date(object sender, EventArgs e) { // Print the date using DateTime "D" format Debug.↩ Log("It is " + last_tick.DayOfWeek.ToString() + ", " + last_tick.ToString("↩ D")); }`

How to add an event to the event stack:
`event_stack.Add(`**`DateTimeEvent.InstantiateDateTimeEvent`** (p. 14)`(start_date.↩ AddSeconds(1), myFunctionToCall));`

How to create your own custom repeating events. // 1) Create your custom ticking event handler `public event EventHandler tick_custom_function;` // 2) Create your custom tick function `protected virtual void OnCustomTick() { tick_custom_function?.Invoke(this, null); }` // 3) Subscribe some function to your custom ticking event handler `tick_custom_function += a_custom↩ _function_to_tick;` // 4) Call your custom tick function somewhere in the pass_time loop. You may call it from another ticking function so that it is not tested every update loop.

```
// For example, if you need to call an event on the 10th day of every month, test it every day by adding it to
```

```
// Called every day protected virtual void OnTickDay(DateTime d) (p. 19) {
tick_day?.Invoke(this, null); if (d.day == 10) { OnCustomTick(); }
```

3.5.2 Member Function Documentation

3.5.2.1 AddEventToStack() [1/2]

```
void TimeManager.AddEventToStack (
              DateTimeEvent e )
```

Adds the **DateTimeEvent** (p. 14) e to the stack.

Parameters

e	

3.5.2.2 AddEventToStack() [2/2]

```
void TimeManager.AddEventToStack (
              DateTimeEvent e,
              CalendarController c,
             DAY_STYLE d )
```

Adds the **DateTimeEvent** (p. 14) e to the stack and highlights the date on a calendar.

Parameters

e	

3.5.2.3 OnTickDay()

```
virtual void TimeManager.OnTickDay (
            DateTime d )  [protected], [virtual]
```

Event called every day.

Parameters

b	

3.5.2.4 OnTickHour()

```
virtual void TimeManager.OnTickHour (
            DateTime d )  [protected], [virtual]
```

Event called every hour.

3.5.2.5 OnTickMinute()

```
virtual void TimeManager.OnTickMinute ( )  [protected], [virtual]
```

Event called every minute. This will be called 1440 times every simulated day.

3.5.2.6 OnTickMonth()

```
virtual void TimeManager.OnTickMonth (
            DateTime d )  [protected], [virtual]
```

Event called every month.

3.5.2.7 OnTickNamedDay()

```
virtual void TimeManager.OnTickNamedDay (
            DateTime d )  [protected], [virtual]
```

Fires events based on what day of the week it is ie. sunday monday tuesday ect.

Parameters

b	

3.5.2.8 OnTickQuarterDay()

```
virtual void TimeManager.OnTickQuarterDay (
            DateTime d )  [protected], [virtual]
```

Tick the quarter day events(12am, 6am, 12pm, 6pm)

3.5.2.9 OnTickQuarterHour()

```
virtual void TimeManager.OnTickQuarterHour ( )  [protected], [virtual]
```

Event called every 15 miuntes.

Parameters

b	

3.5.2.10 OnTickYear()

```
virtual void TimeManager.OnTickYear (
            DateTime d )  [protected], [virtual]
```

Event called every year.

3.5.3 Member Data Documentation

3.5.3.1 game_time_multiplier

```
float TimeManager.game_time_multiplier
```

The game time multiplier.

Multiplied by Time.deltaTime to drive the simulation forward.

Can be set to 0 to pause the simulation.

3.5.3.2 start_day

```
int TimeManager.start_day = 14
```

The sim start day (1 $<$ start_day $<$ Days in month)

3.5.3.3 start_hour

```
int TimeManager.start_hour = 8
```

The sim start hour (0 $<$ start_hour $<$ 23)

3.5.3.4 start_minute

```
int TimeManager.start_minute = 0
```

The sim start minute (0 $<$ start_minute $<$ 59)

3.5.3.5 start_month

```
int TimeManager.start_month = 3
```

The sim start month (1 $<$ start_month $<$ 12).

3.5.3.6 start_now

```
bool TimeManager.start_now = false
```

Set to true to intialize the **TimeManager** (p. 17) with DateTime.Now

3.5.3.7 start_year

```
int TimeManager.start_year = 1762
```

The sim start year. (0 AD $<$ start_year $<$ 9999 AD)

3.5.4 Property Documentation

3.5.4.1 date

```
DateTime TimeManager.date  [get], [set]
```

The current date. This value is incremented using Time.DeltaTime, so it is not gaurenteed to be at any specific time.

3.5.4.2 event_stack

```
SortedSet< DateTimeEvent> TimeManager.event_stack  [get]
```

The event stack is a sorted set based on the date. Automatically stores DateTimeEvents in a sorted list based on the date each event should fire..

The pass_time loop evaluates if the smallest datetime is past the current datetime and fires events accordingly.

Events are removed once they have been fired.

```
// Add a new datetime event to the stack, 1 day after of the current simulated
date.  event_stack.Add(DateTimeEvent.DateTimeEventClass.InstantiateDate↩
TimeEvent(date.AddDays(1), test_event));
```

3.5.4.3 start_date

```
DateTime TimeManager.start_date  [get]
```

The start date of the simulation set by the start_year, start_month, start_day, start_hour and start_minute.

The start date is init'd on start

The documentation for this class was generated from the following file:

- TimeManager.cs

Index