Time Machine

By Jonas Erlinghagen (Avaleryon.com)

-Inspector and Script Documentation-

Table of Contents

1. Getting Started

1.1 Changelog - v.1.1

All:

- Renamed “Date Alarms” to “Date Events” in all Scripts.

TimeMachine:

- Fixed list of Event coroutines not being cleared properly.
- Changed how Events are processed so that it is **now possible to have multiple Events per Date.**
- Changed return type of **setEventAndExecute** method from void to int. The method now returns the index of the newly set Event. This index can be used to cancel a certain event if multiple events have been set for a specific date.
- Changed **setEventAndExecute** method so that it now executes an Event immediately should the new Event’s Date be equal to the current Date.
  - The method will return *“999999”* as an index in that case since no index can be given.
- Changed **cancelEvent** method to cancel a specific Event that corresponds to a given index (int) and was set for a given date.
- Added **cancelEvents** method that cancels all Events set for a given date.
- Added **cancelAllEvents** method that cancels all currently active Events.
- Changed **saveAllTimeData** and **deleteAllTimeData** to take a string **"saveID"** as parameter
  - New save path: *“PersistentDataPath/TimeMachine/Timedata_saveID.tidat”*
- Added **loadAllTimeData(string saveID)** method that loads Time Data from *“PersistentDataPath/TimeMachine/Timedata_saveID.tidat”*
  => TimeMachine no longer loads time data automatically when initialized.
  The method **loadAllTimeData** needs to be called now.
  => There can now be any number of different sets of time data.
- Removed Boolean **istimedataloaded** since it’s no longer needed with manual loading now implemented.

Inspector:

- Fixed errors caused by setting or cancelling Events using the inspector while in Playmode.

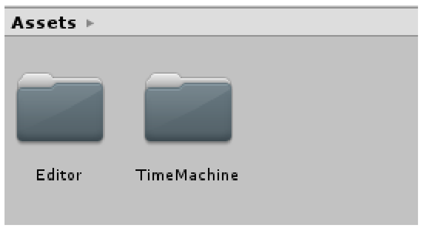
Demo:

- Added “Cancel Events” Button to the GUI.
- Added “Delete Data” Button to the GUI

This Documentation has been updated accordingly.

1.2 Setting up the Asset

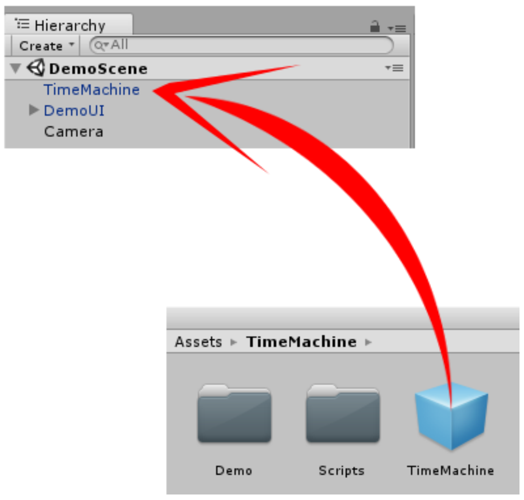
In order to get started using TimeMachine first of all you'll have to move both folders included in the "TimeMachinePackage" folder ("Editor" and "TimeMachine") to the root directory of your Project ("Assets"). Your root directory should now look something like this:



Afterwards you can delete the empty "TimeMachinePackage" folder.

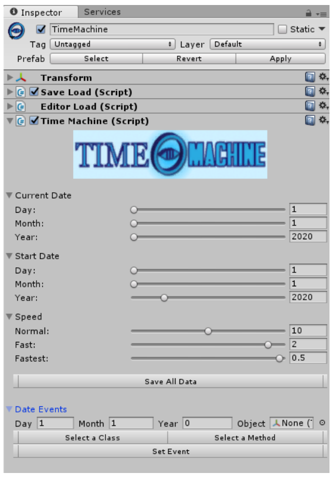
Now you should add the TimeMachine Prefab to your hierarchy.

You can find the Prefab in the TimeMachine Folder.



In your Script you should then reference the TimeMachine Script by getting the Script Component from the Prefab.

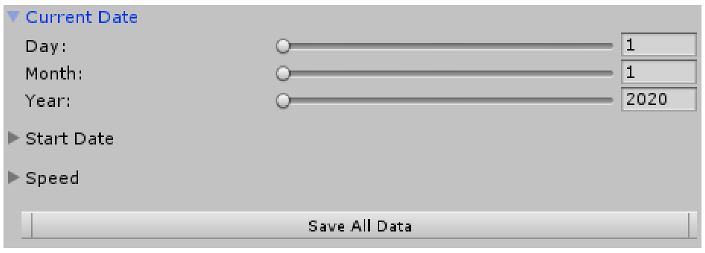
2. The Inspector


Inspector | Services
TimeMachine | Static
Tag: Untagged | Layer: Default
Prefab: Select | Revert | Apply
Transform
Save Load (Script)
Editor Load (Script)
Time Machine (Script)

Current Date
Day: 1
Month: 1
Year: 2020
Start Date
Day: 1
Month: 1
Year: 2020
Speed
Normal: 10
Fast: 2
Fastest: 0.5
Save All Data
Date Events
Day 1 | Month 1 | Year 0 | Object None (
Select a Class | Select a Method
Set Event

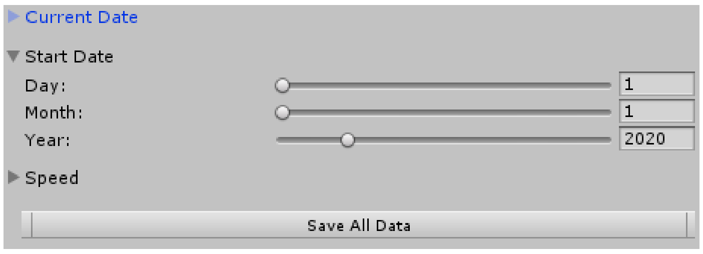
2.1 Current Date

Shows the current date. By adjusting the sliders the current date can be changed.


Current Date
Day: 1
Month: 1
Year: 2020
Start Date
Speed
Save All Data

2.2 Start Date

Shows the start date. By adjusting the sliders the start date can be changed.

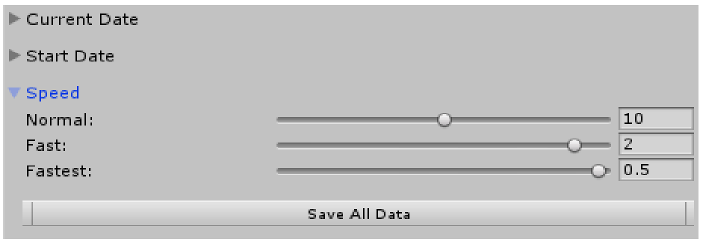

Current Date
Start Date
Day: 1
Month: 1
Year: 2020
Speed
Save All Data

2.3 Speed

Shows the three speed variables which determine how quickly time flows.

The different speed variables can be adjusted with the sliders.

A *smaller* value indicates a *higher* speed.


Current Date
Start Date
Speed
Normal: 10
Fast: 2
Fastest: 0.5
Save All Data

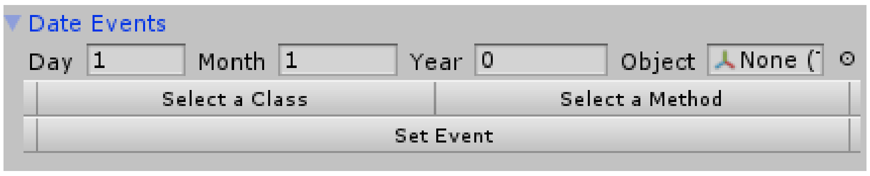
Important:

Remember to press the “Save All Data” button in order to save all values you have changed.

2.4 Date Events

Using Date Events you can execute any number of void methods at any given date.

Date Events can not only be set via script but also via the inspector.

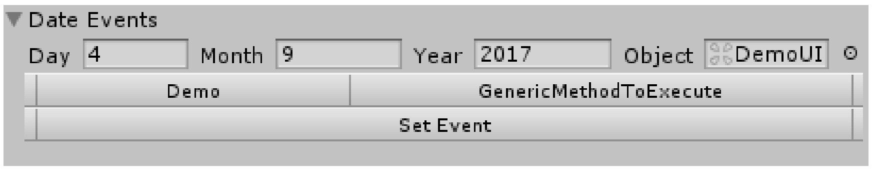


Date Events

Day 1 Month 1 Year 0 Object None (

To do so simply follow these steps:

1. Set the day, month and year according to when you want the Event to trigger.
2. Select the object to which the method you want the Event to trigger is attached.
3. Select the class which contains the method you want the Event to trigger.
4. Select the method you want the Event to trigger.

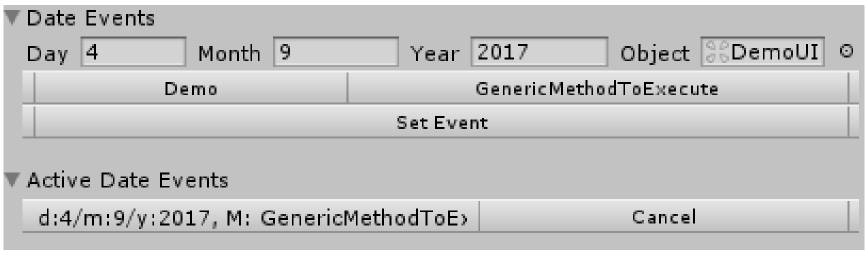


Date Events

Day 4 Month 9 Year 2017 Object DemoUI

Demo | GenericMethodToExecute

5. Now click on the "Set Event" button.



Date Events

Day 4 Month 9 Year 2017 Object DemoUI

Demo | GenericMethodToExecute

Active Date Events

d:4/m:9/y:2017, M: GenericMethodToE›

Your Date Event is now set and will be loaded as soon as you enter Playmode.

The chosen method will then be called at the chosen date.

If you wish to cancel one of your Events simply hit the "Cancel" button.

Be aware that Events that have been set via the Inspector outside of Playmode and are then canceled while in Playmode will not be permanently deleted.

Important:

Be aware that Events set via the Inspector will only be loaded if you didn't load any saved time data using the loadAllTimeData method since this saved data might already contain the same Inspector Events.

3. Script Reference

3.1 Date – Point in time vs. timespan

The Date variable consists of three floats:

```
Date d = new Date(float1, float2, float3);
```

float1 represents days,

float2 represents months,

float3 represents years.

The Date variable can be used to describe a certain point in time, as well as a timespan:

Date(1, 3, 2000)	*could represent either a Date*	*(Day 1 of Month 3 of the Year 2000)*
	or it could represent a timespan	*(1 Day, 3 Months, 2000 Years)*

Please refer to a method's description or this manual if you are uncertain how to use the Date variable with a certain method.

3.2 Date Manipulation

setDate(Date date)

Sets the current date to a given date.

setDateToStartDate()

Resets the current date to the start date.

addLeapYear(int leapyear) /// addLeapYear(int[] leapyears)

Adds additional leapyear(int) or leapyears(int[]) to TimeMachine's array of leapyears.

By default the array contains all leap years from *1804* to *2400*.

renameMonths(string[] months)

Renames month i+1 to months[i] (e.g. months[0] = "Randoary" will rename January to Randoary)

3.3 Time Manipulation

stopStartTime()

Toggle stop or start Time.

stopTime()

Stops Time.

startTime()

Starts Time.

setSpeedToNormal()

Sets the current speed to Speed_Normal.

setSpeedToFast()

Sets the current speed to Speed_Fast.

setSpeedToFastest()

Sets the current speed to Speed_Fastest.

reverseTime(bool doreverse)

Reverses time flow if doreverse == true.

If the Date reaches the StartDate while time is reversing, the time flow will stop.

3.4 Date Information

getDate()

Returns the current date.

getDateAsString()

Returns the current date as a string (e.g. "7 April 2020").

getMonthAsString(int month)

*Returns the name(string) of a given month(**int**) (e.g. 1 == "January").*

isLeapYear(int year)

Checks if the given year is a leap year.

getDaysInMonth(float month, float year)

Returns the number of days in a given month of a given year
(Also takes leap years into account).

3.5 Date Calculation

getDateAndTimeSum(Date date, Date timetoaddtodate)

Returns the sum of a Date(Date) and a Date(Timespan)
(Also takes leap years into account).

The first variable (date) represents an actual date
(e.g. Date(28, 9, 2016) – The 28th of September 2016),
the second (timetoaddtodate) represents the amount of time you want to add to the aforementioned date.
(e.g. timetoaddtodate = Date(0, 5, 1) will add 0 Days, 5 Months and 1 Year to the given date(date))

getTimeDateInDays(Date timespan, Date reference)

Returns a given timespan of the Format Date(days, months, years) in days as a float.
A reference point (reference) is needed for an accurate calculation (Date as Point in Time).

getMultipliedTimespan(Date timespan, Date reference, float multiplier)

Multiplies a timespan of the format Date(days, months, years) and returns the result in days as a float.
A reference point (reference) is needed for an accurate calculation (Date as Point in Time).

getTimeDaysAsDate(float days, Date startdate)

Takes a given number of days (days) and calculates the exact number of months, years and rest-days and returns them as a timespan of the format Date(days, months, years).
A reference point (startdate) is needed for an accurate calculation (Date as Point in Time).

3.6 Date Events

setEventAndExecute(Date date, Action methodtoexecute)

Executes a given method(methodtoexecute) at a given date.

Example:

```
TimeMachine tm;                        |TimeMachine reference|
Date date = new Date(1, 3, 2021);      |The Date when we want our Method to execute|
                 =>Date(day, month, year)<=
public void GenericMethod(){}          |The Method we want to execute at a certain date|

tm.SetEventAndExecute(date, GenericMethod);
|This would execute GenericMethod() on the 1.3.2021 (first of march 2021)|
```

Also returns the index (int) that has been assigned to the new Event. You can use this index to cancel the Event using the cancelEvent(Date date, int index) method.

cancelEvent(Date date, int index)

Cancels the specific Event with index index that was set for the given date.

cancelEvents(Date date)

*Cancels **all** Events that were set for the given date.*

cancelAllEvents()

*Cancels **all** currently active Events.*

3.7 Time Data

saveAllTimeData(string saveID)

Saves all Time Data (including active Date Events) to
*'PersistentDataPath/TimeMachine/Timedata_**saveID**.tidat'.*

deleteAllTimeData(string saveID)

Deletes all saved Time Data (including active Date Events) located at
*'PersistentDataPath/TimeMachine/Timedata_**saveID**.tidat'.*

loadAllTimeData(string saveID)

Loads all Time Data (including active Date Events) from
*'PersistentDataPath/TimeMachine/Timedata_**saveID**.tidat'.*
