Event Editor

| **Customer** | Alexander Cvetanovic |
| --- | --- |
| **Project Manager** | Alexander Cvetanovic |
| **Autor** | Alexander Cvetanovic |
| **Classification** | Home/Student Project |
| **Status** | Work in Progress |
|  |  |

Version Control

|  |  |  |
| --- | --- | --- |
| Version | Date | Change |
| 0.1 | 31.09.19 | Document created |
| 0.2 | 05.09.19 | Added Structure of Hermes; Formatting the documentation |
| 0.3 | 24.11.19 | Added Secondary Task, completed Object Analysis and Solution Search. |

Table of Content

[Version Control 2](#_Toc25513391)

[Table of Content 3](#_Toc25513392)

[1 Initiation 5](#_Toc25513393)

[1.1 Situation 5](#_Toc25513394)

[1.2 Project Goals 5](#_Toc25513395)

[1.3 Personal Goals 5](#_Toc25513396)

[1.4 Task Definition 6](#_Toc25513397)

[1.4.1 Unified Handling 6](#_Toc25513398)

[1.4.2 Event Like Behavior 6](#_Toc25513399)

[1.4.3 Primary Tasks 6](#_Toc25513400)

[1.4.4 Secondary Tasks 6](#_Toc25513401)

[1.5 Object Analysis 6](#_Toc25513402)

[1.5.1 Editor Scripting 7](#_Toc25513403)

[1.5.2 Inspector 7](#_Toc25513404)

[1.5.3 Scene Inspector 7](#_Toc25513405)

[1.6 Solution Search 7](#_Toc25513406)

[1.6.1 RPG Makers Event Editor 7](#_Toc25513407)

[1.6.2 Unity Asset Store: TimeLine Events 7](#_Toc25513408)

[1.6.3 Unity Asset Store: Event Manager 7](#_Toc25513409)

[2 Concept 8](#_Toc25513410)

[2.1 Solution 8](#_Toc25513411)

[2.2 Concept of Event Editor Functionality 8](#_Toc25513412)

[2.2.1 Page 9](#_Toc25513413)

[2.2.2 Condition 9](#_Toc25513414)

[2.2.3 Trigger 10](#_Toc25513415)

[2.2.4 Setup 10](#_Toc25513416)

[2.2.5 Functions 10](#_Toc25513417)

[2.3 Concept of Global and Local Switches 11](#_Toc25513418)

[2.4 Visual Design 11](#_Toc25513419)

[2.4.1 Inspector 11](#_Toc25513420)

[2.4.2 Scene View 11](#_Toc25513421)

[2.5 Test Concepts 12](#_Toc25513422)

[3 Realization 13](#_Toc25513423)

[3.1 Milestones 13](#_Toc25513424)

[3.2 Time 13](#_Toc25513425)

[3.3 Realization of Global Switches 14](#_Toc25513426)

[3.3.1 Code 14](#_Toc25513427)

[3.4 Realization of the Event Editor 14](#_Toc25513428)

[3.5 Realization of Tests 14](#_Toc25513429)

[4 Introduction 15](#_Toc25513430)

[4.1 Setup of an Event 15](#_Toc25513431)

# Initiation

## Situation

Unity is a free engine to create games with. Unity’s code to object relation is a composition. A game object has mono behaviors in form of attached components. This enables many variations of interaction between other objects and the engine itself. This makes Unity easy to use for a broad genre of games.

This “openness” however has two sides to it. Unity does not have specific systems for a specific genre. Good thing here that we have Asset Stores, which cover this problem for us for most of the cases. But there is still a lot of potential left.

One such needed potential is a tool designed for adventure like RPG that simplifies the handling with events and objects.

## Project Goals

* Project finishes on deadline for the Homework which is the 18.12.19.
* New easier way to work with game objects.
* Simple setup for events

## Personal Goals

* Clear distinction between Game objects with or without Events.
* Simple code using a modular structure to extend features in the near future.

## Task Definition

### Unified Handling

All events have to be same in their functionality. This is very important as they are a lot of events inside the scene.

### Event Like Behavior

For an RPG adventure like games, objects have to be interactable and aware with their surroundings and mostly with the player himself. These objects have to be clarified by the new Tool to act as events.

### Primary Tasks

Table 1 Primary Task List

|  |  |
| --- | --- |
| Code | Description |
| T001 | Add, Remove or Modify list of pages on the editor. |
| T002 | Selecting a Page updates the Editor. |
| UC01 | Add, Remove or Modify list of conditions on the editor |
| UC04 | Set a Condition from a Scene Object and an exposed variable. |
| UC05 | Bool conditions test if the Variable is either like or not like the other bool variable |
| UC06 | Integer/Float conditions test if the Variable is either same, not same, greater than, greater or equal than, lesser than or lesser or equal than the other integer variable or a set integer value. |
| UC07 | String conditions test if the Variable is either same or not same as the other string variable or a string value. |
| UT01 | Set Trigger to: Autorun, Parallel or Interact. |
| UT02 | Interact trigger triggers on Interaction calls from other objects. |
| UT03 | Interact trigger triggers on Collision or Trigger calls by other objects. |
| UT04 | Autorun trigger triggers automatically when the object has been loaded in the scene. |
| UT05 | Parallel trigger triggers automatically when the object has been loaded in the scene and loops endlessly until the event or page is no longer active. |
| US01 | Setup the default sprite. |
| US02 | Setup the animator and initial animator Speed. |
| US03 | Setup the default Character State Controller. |
| US04 | Setup Sorting Layer. |
| UF01 | Add, Remove or Modify Unity Events in the function list on the editor. |

### Secondary Tasks

Table 2 Secondary Task List

|  |  |
| --- | --- |
| Code | Description |
| T003 | It works perfectly with the Unity Undo System. |
| T004 | In Scene View highlight objects with an Event Editor. |
| UC02 | Set a Condition with a Global Switch |
| UC03 | Set a Condition with a Local Switch |
| UC08 | Local Switch condition test if a local Switch is true. |
| UC09 | Global Switch condition test if a global Switch is true. |
| UC10 | Global Switches can be Added, Removed or Modified its own Global Switch Window. |
| UC11 | Conditional Fields from Objects can take 1 Parameter type of either bool, int, string, Game object, Vector2, Vector3 |

## Object Analysis

### Editor Scripting

This enables the editor to have custom editorial windows for the user. This project will focus on these scripts. Editor scripting can change the entire application and gives the user so much more customizable options.

### Inspector

This project will most likely benefit from a custom inspector script. Since the Event Editor is attached on a game object the setup its values are then shown in the inspector. The Inspector can be modified by an Editor script.

There is a large amount of layout elements available, which can be used to create a functionable Event Editor. However, most important is the reordable list. This list is Unity Editor internal and does not have a public API. This makes it difficult to implement, but not impossible.

### Scene Inspector

The scene inspector displays all active objects in the current scene. It also displays debug and transformation elements as well. What’s interesting is that it has a functionality to highlight certain objects depending on what category is searched by.

## Solution Search

There are solutions existing, which can completely satisfy the conditions of an Event Editor, despite it being a too specific solution.

### RPG Makers Event Editor

There is an engine out there called the RPG Maker. The RPG Maker series are all known for their simplicity because their Development Kit requires no knowledge of coding.

### Unity Asset Store: TimeLine Events

There exists an asset that creates a timeline in which function are then executed by. It is primary made for animations.

### Unity Asset Store: Event Manager

This plugin enables the user to register functions to already made event handlers. It can take functions with parameters and is able to test triggering from the configuration window.

# Concept

## Solution

This Tool enables game objects to have an event editor in their inspector. This event editor makes game objects easier to handle, because all objects are handled in a uniform matter.

## Concept of Event Editor Functionality

The event editor handles its game object based on the settings done in the event inspector. Features are:

* Pages: Each event can have one or many pages.
* Conditions: Determines whether a page is active or not.
* Trigger: Determines how the event can be interacted with.
* Processing: Determines how the event is processed when it is active.
* Setup: Each Page has a setup section which sets up the game object.
* Functions: This is a list of function which are called when the event is triggered.

The following illustration displays what the user can change directly with the new event editor on the game object.

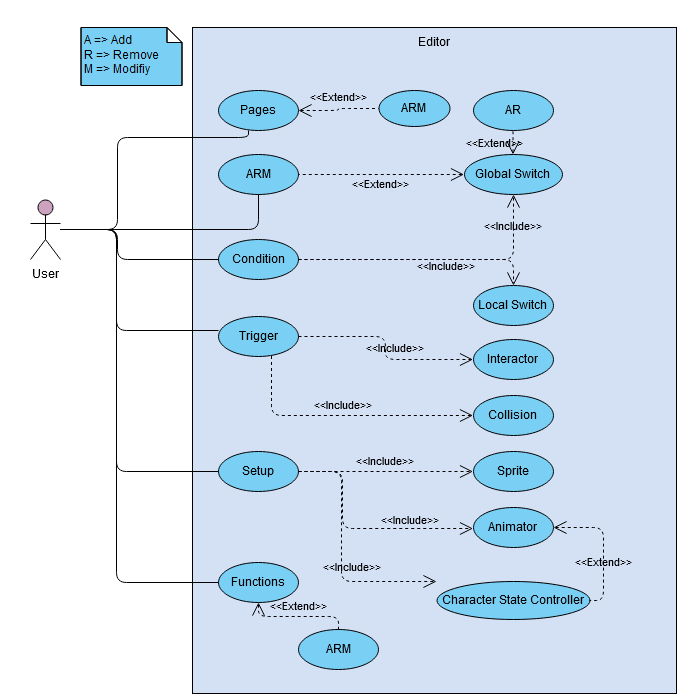


Figure 1 Use Case Diagram of the Event Editor

### Page

An event can have one or multiple pages set up in the inspector. Each page has its own set of values, which can change the behavior of its game object. Only one Page can be active at a time. This depends which page first meets their condition. Pages can be added, copied or removed.

### Condition

Each Page has a condition section. The first page which all conditions are met is active for the event. A page can have none to many conditions. Following conditions can be set up in the inspector:

* Global Switch: This is a global dictionary of type <string, bool> value. This dictionary can be accessed anywhere and anytime. It is stored on the hard disk.
* Local Switch: This switch exists only on the event itself and on all its pages. It cannot survive scene changes.
* Game object variables: All specified variables exposed with the tag “[ConditionalField]” are listed from the specified game object found in the same scene. Depending on the type of the variable the condition can be set as following:
  + Integer/Float: variable is **equal**, **greater than**, **greater or equal than**, **less than**, **less or equal than** or **not** like the other variable or value
  + Bool: variable is either **same** or **not like** thecompared to the other variable or value.
  + String: variable is either **same** or **not** like the other variable or value.

### Trigger

This determines how the event can be interacted with. There are:

* Parallel: Automatically processes function calls in the function section in a coroutine and loops when the list is finished.
* Autorun: Automatically processes function calls in the function section in a coroutine.
* Interaction: Triggers only on interaction calls from an Interactor.
* Collision: Triggers on collision controls. **OnTriggerEtc.** or **OnCollsionEtc.**

### Setup

These changes setup the event from the moment where when a page becomes active. Changes are:

* Sprite: What sprite is shown.
* Animator: Which animator to use.
* Character State: **IsDirectionFix**, **DefaultState**
* Sorting Layer: Sets Sorting Layer for the object.
* Animation Speed: How fast the animator plays animations.

### Functions

This list of function is called when the event is being triggered. It’s essentially a UnityEvent list where all objects inside the scene can subscribe their functions to.

In later iterations this list might get revamped to or completely replaced by an own Custom Event List, which has new inspector elements for better functionality. These elements can then be attached to one and other. The following Mini Tools will be a part of this new list.

* Flow Control: Conditional Branches, Loops, etc.
* In-depth Object Control: Custom tools for managing object variables like health, buffs, inventory, Camera Control, Shaders, Map changes, Tile Set handling etc.
* Game Control: Custom tools for managing game flow like battle processing, Scene management, etc.
* Message integration: A custom message tool to create dialogs.

## Concept of Global and Local Switches

// Explain what this is and how it works.

## Visual Design

### Event Editor

// Mockup of the Event Editor Inspector

### Global Switches

// Mockup of the Global Switches window

### Scene View

// Mockup of the Scene View with Editor highlight.

## Test Concepts

// To be made.

# Realization

## Milestones

Table 3 Milestone List

|  |  |  |
| --- | --- | --- |
| Id | Date | Name |
| M1 | 22.09.19 | Project Start and Task order |
| M2 | 06.10.19 | End of the initial phase |
| M3 | 13.11.10 | End of the concept phase |
| M4 | 10.12.12 | End of the realization phase |
| M5 | 18.12.10 | Project finish |
|  |  |  |

## Time

// Time table

## Realization of Global Switches

### Code

// Code Model

// Show Picture of the new window

## Realization of the Event Editor

// Code Model

// Show Picture of the new inspector.

## Realization of Tests

// Test your stuff.

# Introduction

## Setup of an Event

// Explain like what sets the user takes to create a new event

// Explain functionality of every button and field on the inspector.