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. |
| 0.3a | 28.11.19 | Addendum to the last version. Added Condition color codes. |
| 0.4 | 04.12.19 | Finalized changes in the project and updated the document. |

Table of Content

[Version Control 2](#_Toc25936494)

[Table of Content 3](#_Toc25936495)

[1 Initiation 5](#_Toc25936496)

[1.1 Situation 5](#_Toc25936497)

[1.2 Project Goals 5](#_Toc25936498)

[1.3 Personal Goals 5](#_Toc25936499)

[1.4 Task Definition 6](#_Toc25936500)

[1.4.1 Unified Handling 6](#_Toc25936501)

[1.4.2 Event Like Behavior 6](#_Toc25936502)

[1.4.3 Primary Tasks 6](#_Toc25936503)

[1.4.4 Secondary Tasks 6](#_Toc25936504)

[1.5 Object Analysis 7](#_Toc25936505)

[1.5.1 Editor Scripting 7](#_Toc25936506)

[1.5.2 Inspector 7](#_Toc25936507)

[1.5.3 Scene Inspector 7](#_Toc25936508)

[1.6 Solution Search 8](#_Toc25936509)

[1.6.1 RPG Makers Event Editor 8](#_Toc25936510)

[1.6.2 Unity Asset Store: TimeLine Events 8](#_Toc25936511)

[1.6.3 Unity Asset Store: Event Manager 8](#_Toc25936512)

[1.7 Solution 8](#_Toc25936513)

[2 Concept 9](#_Toc25936514)

[2.1 Core Concept of the Event Editor 9](#_Toc25936515)

[2.2 Event Editor Feature Modell 10](#_Toc25936516)

[2.2.1 Page 11](#_Toc25936517)

[2.2.2 Condition 11](#_Toc25936518)

[2.2.3 Setup 11](#_Toc25936519)

[2.2.4 Trigger 11](#_Toc25936520)

[2.2.5 Functions 12](#_Toc25936521)

[2.3 Core Concept of Global Switches 13](#_Toc25936522)

[2.4 Global Switch Feature Model 13](#_Toc25936523)

[2.5 Visual Design 13](#_Toc25936524)

[2.5.1 Event Editor 13](#_Toc25936525)

[2.5.2 Global Switches 13](#_Toc25936526)

[2.5.3 Scene View 13](#_Toc25936527)

[2.6 Test Concepts 14](#_Toc25936528)

[3 Realization 15](#_Toc25936529)

[3.1 Milestones 15](#_Toc25936530)

[3.2 Time 15](#_Toc25936531)

[3.3 Realization of the Event Editor 16](#_Toc25936532)

[3.4 Realization of Global Switches 16](#_Toc25936533)

[3.4.1 Code 16](#_Toc25936534)

[3.5 Realization of Tests 16](#_Toc25936535)

[4 Introduction 17](#_Toc25936536)

[4.1 Event Editor Manual 17](#_Toc25936537)

# Initiation

This Chapter covers the initiation phase of the project.

## 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 homework deadline which is the 18.12.19.
* New workflow that handles game objects similarly across a the project.
* New Component for game objects to setup a Game Event.

## 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: Interact, Autorun or Custom. |
| UT02 | Interact Trigger: Function call from the Game Event. |
| UT03 | Custom Trigger: checks for a custom trigger implemented with an interface in other Components. |
| UT04 | Autorun triggers immediately, after the page becomes active. |
| 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 with the Unity Undo System. |
| T004 | In Scene View highlight objects with an Event Editor. |
| T005 | Online API for the Tool. |
| US01 | Setup the default sprite. |
| US02 | Setup the animator and initial animator Speed. |
| US03 | Setup the default Character State Controller. |
| US04 | Setup Sorting Layer. |
| 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 |
| UC12 | Conditional Fields are color-coded depending on their status: ORANGE -> Condition not Valid; RED -> Condition Check returns FALSE; GREEN -> Condition check returns TRUE; BLUE -> Marks active page during runtime. |

## 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. Their engine is, and quoting, “simple for beginners and powerful for developers”. All this is thanks to their Event Editor. This tool enables the user to set up events on the map to interact differently with the game world.

### 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.

## Solution

The Editor will be created from scratch with the inspiration of the Event Editor from the RPG MAKER series. Features and implementation are kept as Unity Native as possible.

# Concept

This Chapter covers the concept phase of the following elements:

* Concept of the Event Editor
* Concept of the Global Switch Manifest

## Core Concept of the Event Editor

The Event Editor is a custom editor for the new component of type GameEvent, which will be added to a game object. The design idea of the Event Editor is that it works with other components inside of the game object and acts like a director, who dictates how function calls are processed.

### User Interfaces and Code Accessablity

The idea of this tool is to give the user a new easier way to work with game objects. Though, GameEvents functionality and logic is mostly done by the Event Editor, the user can control certain aspects of the process, too. It can go so far that GameEvents can be created at runtime by the player.

An online API will be provided with this Tool to help the user find these interfaces of the Event Editor.

### Scope and Handling of Game Events

GameEvents are only active in their active scenes. Inactive Event Editors will not process functionality.

GameEvents are tracked by a global static list. Inside the list are all currently active Event Editors inside active objects of active scenes.

A GameEvent automatically adds to or removes themselves from the list depending on their game object’s active state or whether the Event Editor is active or not. The list refreshes itself every time on scene changes.

## Event Editor Feature Modell

The event editor handles its game object based on the settings done in the event inspector. The Setting are divided in six categories:

* 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.

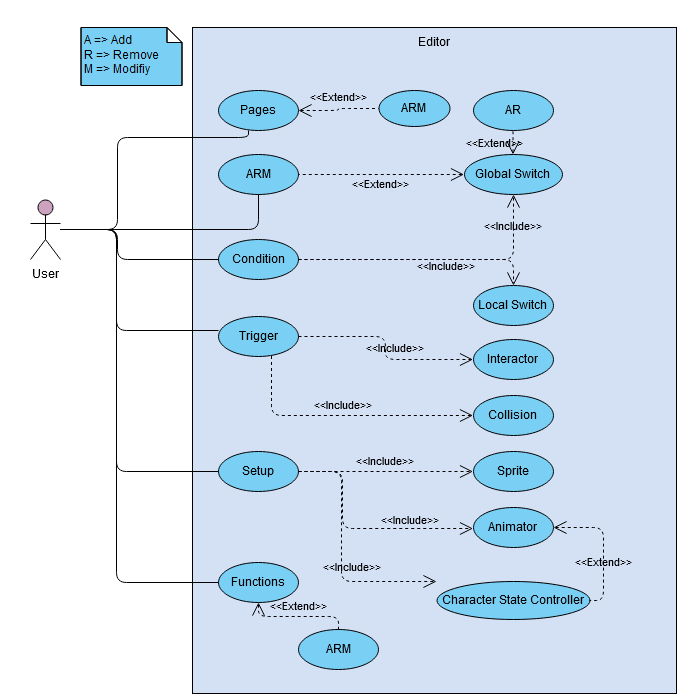


Figure 1 Use Case Diagram of the Event Editor

## 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, reordered or removed.

## Condition

Each Page has a condition section. The first page with all met conditions is considered active for the event. A Page with no conditions can be active as well. 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 a custom extension to the Event Editor. It will be explained in their respective chapter. It is stored on the hard disk.

* More in the chapter about Global Switches.

### Local Switch

Local Switches are stored inside the GameEvents. Their scope doesn’t go beyond their GameEvent and are accessable across all its pages.

A Local Switch is an internal Dictionary of type string and Boolean. The Key and Value of the Dictionary are user defined. It can be created and modified by code and Event Editor.

Local Switches can be defined as conditions, even if the actual local switch doesn’t exist at the moment. This type of condition will always return false. But it changes during runtime as soon as the local Switch (with the same key and positive value) gets created. Then it will return true.

The current list of Local Switches can be accessed on the Event Editor itself via a button.

### Game Object

All specified fields, properties and methods (without parameters) exposed with the attribute “[Conditional]” are listed from the specified game object found in the same scene. Depending on the type and - only these object types - of the variable the condition can be set as following:

* Bool: variable is either **same** or **not like** thecompared to the other variable or value.
* 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
* String: variable is either **same** or **not** like the other variable or value.

## Setup

The Setup section changes parameters of the game object from the moment on the page becomes active. What changes the setup does is heavy depending on the desires of the user.

## Trigger

The Trigger starts the function process of the active page. There are different types of Trigger.

### Interaction

This Trigger triggers on the “Interaction” call made on the GameEvent. Usually this is used for objects that have to process as soon as the player interacts with them.

The Interaction function is accessible on the GameEvent component itself and can be additionally called by its static version, which also takes in the string name of the object the GameEvent is located on.

### Autorun

Automatically triggers after the Setup section is complete.

### Custom

This Trigger looks out for an implemented function from an interface. This Function then checks its implementation and returns the active state of the trigger.

### Run as Coroutine

This is a separate option outside the triggers. Run as Coroutine processes the function invoke as a coroutine, enabling it to run parallel with the gameplay.

## Functions

This list of function is called when the event is being triggered. It’s essentially a Unity Event 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, Breaks, etc.
* In-depth Object Control: Custom tools for managing object variables like health, buffs, inventory, Camera Control, Shaders, Map changes, Tile Set handling, Movement, Character States, etc.
* Game Control: Custom tools for managing game flow like battle processing, Scene management, etc.
* Message integration: A custom message tool to create dialogs.

## Core Concept of Global Switches

Global Switches are an extension tool for the event editor. They keep track of all flags and are stored on the hard drive, essentially making them accessible from any point in the game. This furthers the condition depth and possibility what the event editor can achieve.

## Global Switch Feature Model

// Use Case

## 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

This Chapter covers the realization phase of the project.

## 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 the Event Editor

// Code Model

// Show Picture of the new inspector.

## Realization of Global Switches

### Code

// Code Model

// Show Picture of the new window

## Realization of Tests

// Test your stuff.

# Introduction

## Event Editor Manual

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

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