Intersect Event System Tutorial

Version 1.0

By: Agoraphobic

Welcome, to the ***Intersect: Event System Tutorial***! Intersect uses an advanced *Event System* that you might find similar in other game creation engines; however, *Intersect's Event System* has some core differences that are tailored-made for an ORPG game creation system. If you have worked with *Event Systems* in other engines, this will tutorial will help guide you in adjusting from a single-player, game developer mindset, into an ORPG or MMORPG approach. If you are new to the *Event System*, this tutorial will be essential in learning the ins and outs of *Intersect's Event System*. Working with *Events* is crucial in making a well-rounded ORPG/MMORPG. I promise, that through practice and this tutorial, that working within the *Event System* will become second nature to you.

Overall Goals in Successfully Learning and Using the Event System

- Read through this tutorial and use it as a reference when you need it.
- Apply what you learn by creating your own *Events* and practicing.
- Post questions in the forum, when you are stuck.
- Have fun, and do not get frustrated. You will get it! ^_^

Preface

Writing this tutorial has been a large undertaking; however, it has been my pleasure to write something to help the Intersect Community. My goal with this work is to help our current and future game developers either adapt or learn Intersect's Event System. While it is not completely comprehensive, I hope this tutorial will give you the tools to better understand Intersect and feed your creativity to surpass what this document encompasses. It is also important to bring up why I chose to use a more conversational tone in this piece. My goal was to ensure, no matter the reader's age or experience, this tutorial could be understood and followed with relative ease.

This document will continue to update as the Engine evolves and as I make corrections. Please, feel free to email me or send a message/post on the site with bugs, suggestions, or anything else! Best of luck!

Always Your Friend,

-Agoraphobic

Table of Contents

What are Events?

***Events*:** Think of *Events* as different size boxes with clay within it. There are many sizes of boxes that you can use: small ones, large ones, square ones, round ones, etc. We also can choose what the box is made from: oak, stone, plastic, or any other material! When we say an *Event*, we are simply referencing all of the boxes in general; however, we can get more specific with them! When we create an *Event*, we are doing several things: choosing what type of box it is, what the box is made from, what are we going to put in this box, and of course we are writing its name on it. Just like a toy box, we put toys in it, decorate it like a toybox, and we write that it is a "Toybox" on it. Let us look at an example in game terms.

Event Creation Steps

(1) What type of box is it?
   a. Let's make it a "Person Box" that talks to us!
(2) What is the name of the box?
   a. Let's name the person: "Lady Marie."
(3) Let us decide the material the box is made out of!
   a. Choose the Graphic, Move Route, and other conditions.

We now have created an Event named Lady Marie that is going to a specific person with her own movement and graphics. It is important to note that with this example, we could make many more types of boxes (*Events)*: a "Spike Trap" *Event*, a "Button" *Event*, a "Quest" *Event* and numerous other things that will make our life easier when making our game. The box is completed, but it is still empty! Now we need to look at the "clay" within the box, also known as the Event Commands.

Event Commands: Whenever we create an Event (Box), it is always empty, and we are holding a big block of clay. What we need to do is "break and mold the clay" into several objects.

We are choosing the shapes and forms of the broken clay by molding it in different ways. After we are done, we put it in the box. To clarify what we are talking about in game terms, this is choosing Event commands and putting them inside the Event. When the Event activates, it will run all the Event Commands inside it. Let us look at an example of placing the clay (Event Commands) within the box (Event):

Event Commands Steps

(1) What do we want the "Person" to say?

  a. We choose the "*Show Text"* Command to make Lady Marie say "Hello."

(2) Do we want Lady Marie to do anything else?

  a. We choose *"Play Sound"* to have her play the "Hello.wav"

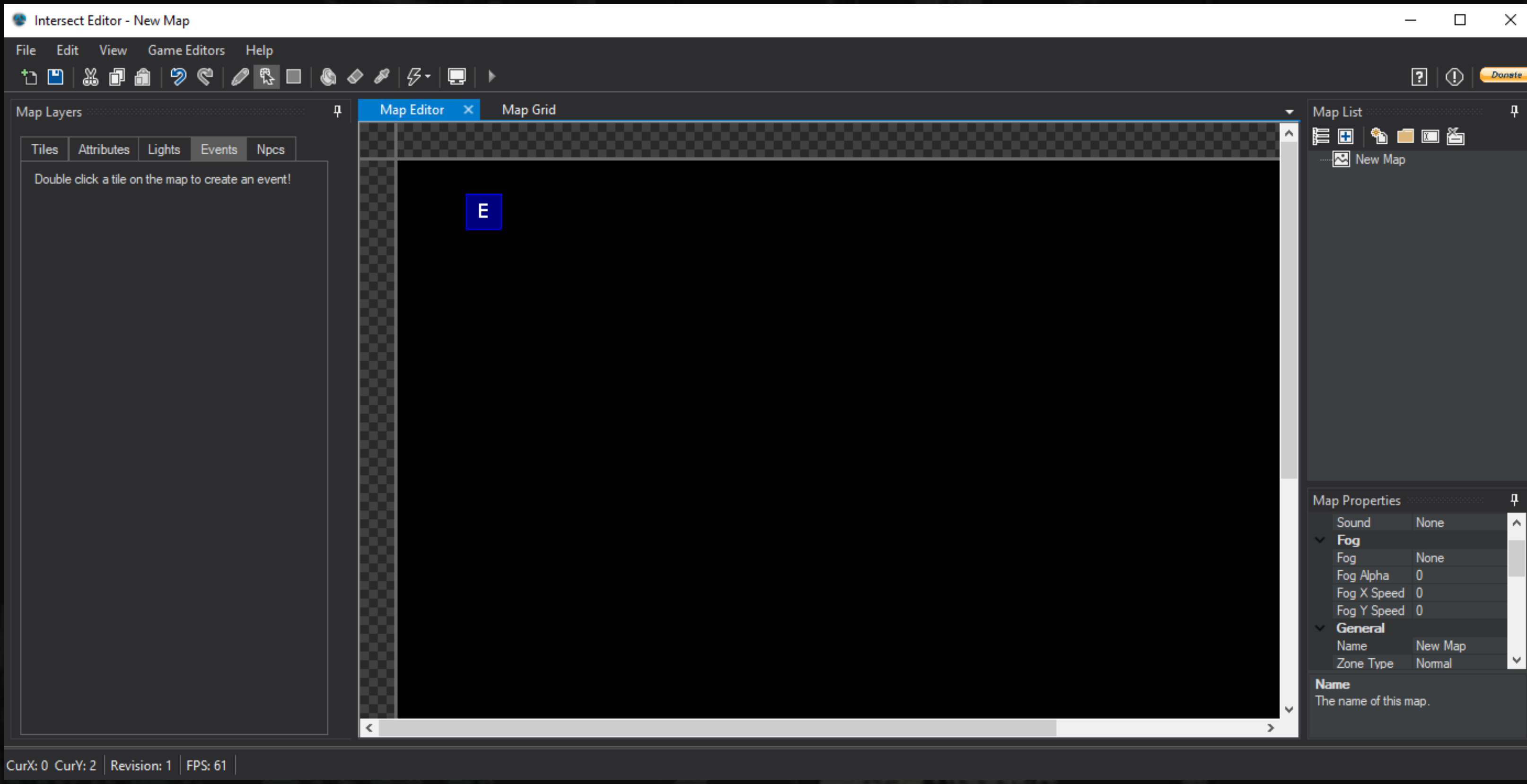
Now, when the player interacts with Lady Marie's, she is going to greet up in both text and sound. This is just one example of how we can create different boxes and put things into them. We created a "Person Box", decorated it, named it, and put objects in it that a "Person Box" could have in it.

As we continue in this tutorial, we are going to go through all the *Event Commands* to show what we can create with *Events*. Hint, there will be quite a bit. You might feel a bit overwhelmed at this point, but do not worry because we will get through it together. This is just to get you familiar with how *Events* and Event Commands work together. When we start going through each *Event Commands* and the structures of the *Event*, it will be clearer with how it all comes together.

Anatomical Structure of an Event

Now that we got all the terminology out of the way, is time for us to learn how to create an Event. In this section, we will be looking at the overall structure of an Event. Then we will also be taking a look at each individual Event Command to gain a better understanding of how it operates on its own and overall within the Event. Most of the Event Commands are straightforward; however, there will also be Events that are more complex than their initial nature suggests. Let us jump right in at looking at how to even create an Event:

Open the Intersect Editor



Double click a tile on the map to create an event!

Sound	None
Fog	
Fog	None
Fog Alpha	0
Fog X Speed	0
Fog Y Speed	0
General	
Name	New Map
Zone Type	Normal

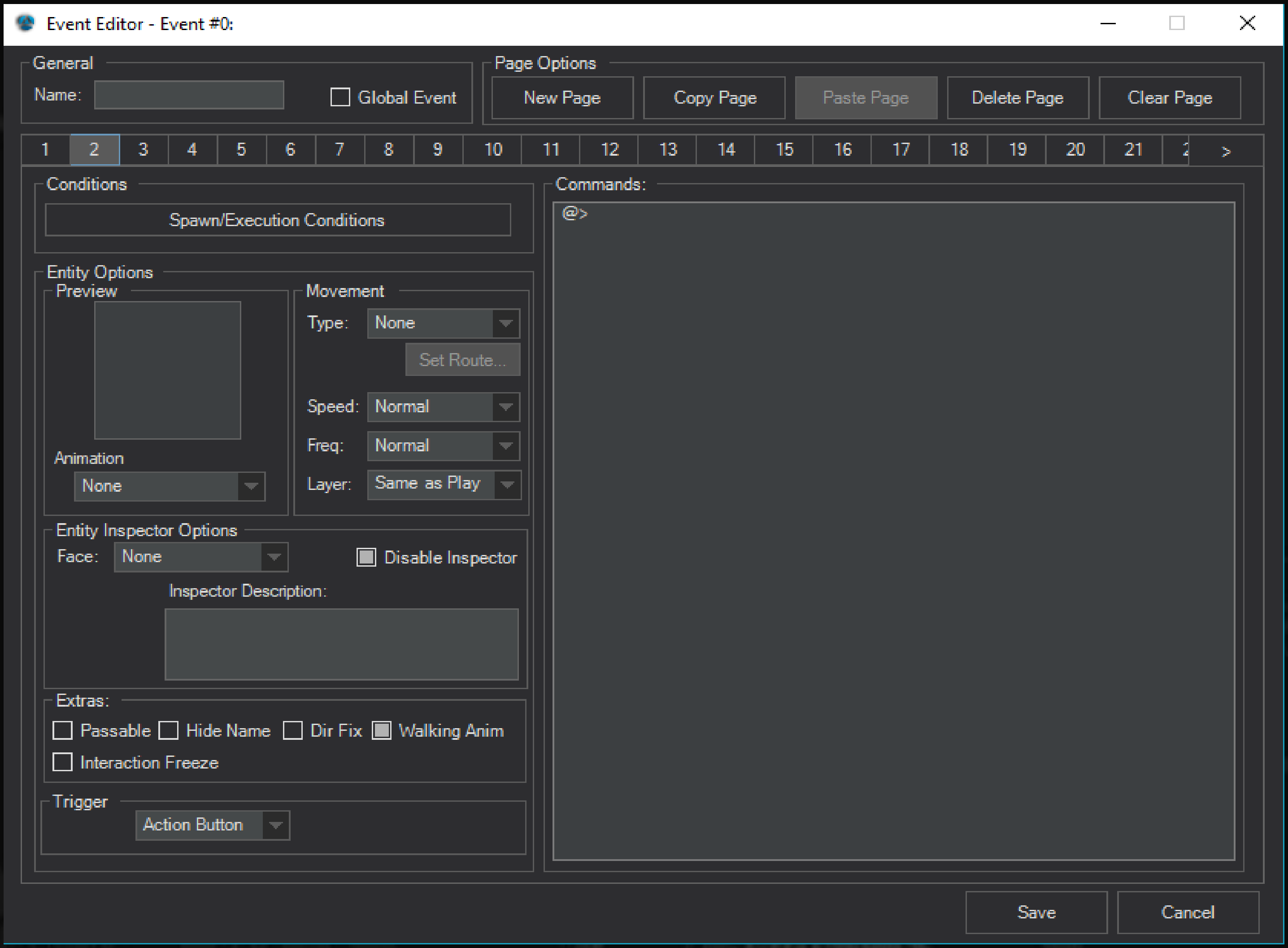
Name
The name of this map.

CurX: 0 CurY: 2 | Revision: 1 | FPS: 61

Click the "Events" tab, and double click on the map to place an Event down. Double clicking a New Event or double clicking a previously placed Event will open the Event Editor for us.

Event Editor

This is what our empty box looks like before we choose anything. Stop, breathe, it will be okay. This may look a bit overwhelming, but we will get through this. The best way to tackle this monster is to divide it up via color coding it!



Name:

Global Event [ ]

Type: None

Speed: Normal

Freq: Normal

Layer: Same as Play

Animation: None

Face: None

Disable Inspector [✓]

Passable [ ] Hide Name [ ] Dir Fix [ ] Walking Anim [✓]

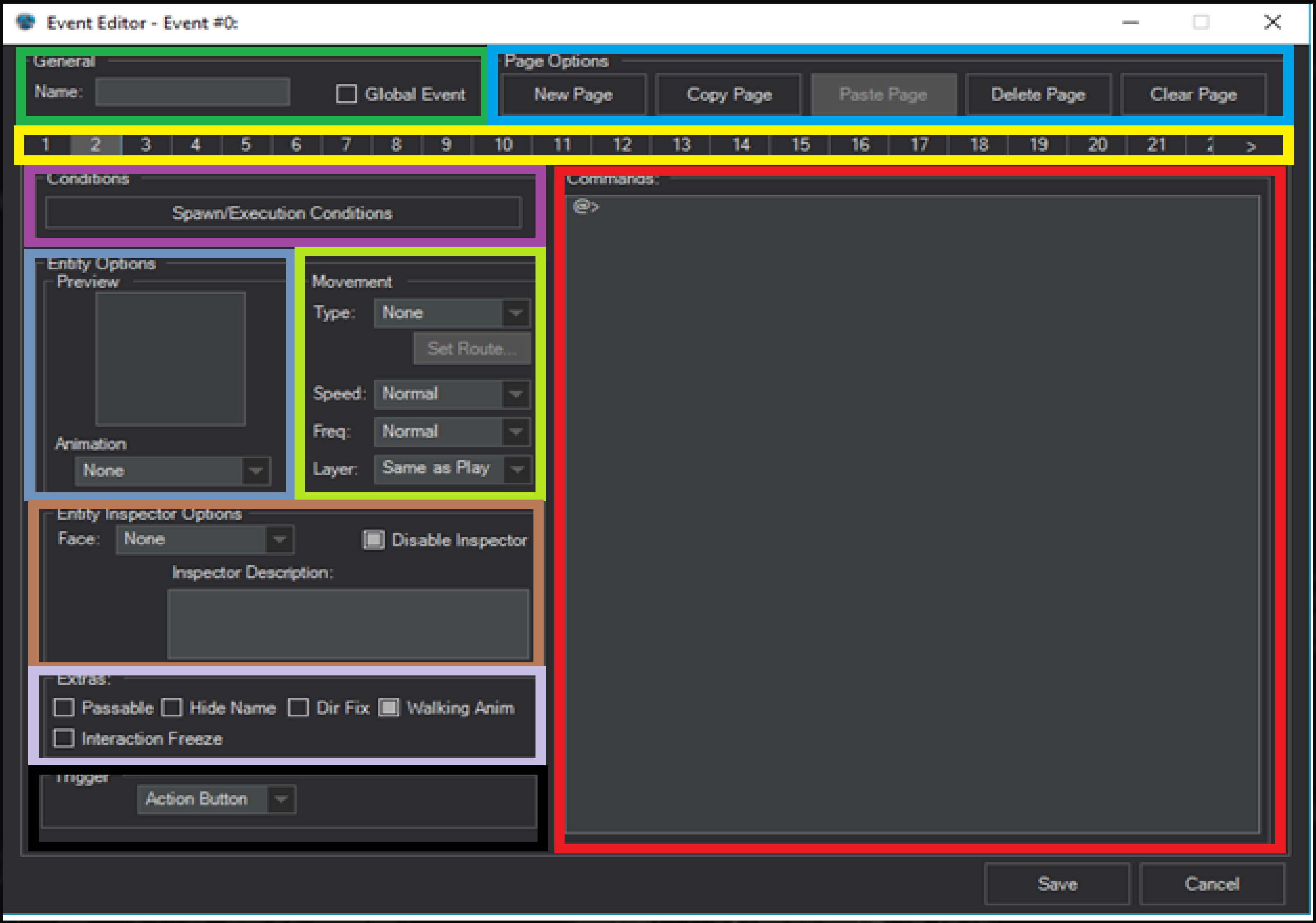
Interaction Freeze [ ]

Trigger: Action Button

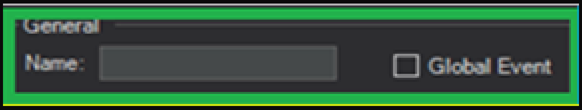
(Note your editor will only show 1 page, not 2-21. Clicking *Add Page* will add more)

Color-Coded Event System

Now, that is much better. Let's go through each colored box and see how each part works within the Event, and examine which color represents the analogy of the box and clay.



General



Name

This is the name of the Event that the engine will reference when using Event Commands. Also, if you enable it, the Player will also see the name hovering over the Event. Because of this, we must not only give the Event a name, but it needs to be good and/or descriptive. By not naming Events or assigning terrible or nonsensical names will cause problems for you in the long run. There are several rules of thumb things you should keep in mind when naming an Event:

(1) Do not leave the name empty.

(2) Do not put random letters, words, numbers, or other weirdness to identify it.

(3) Use Names for characters, numbers after if referenced more than once on a map.

(4) For objects, be descriptive or the general function of it.

Let's look at a few examples of what I mean:

Terrible Name Examples:

"Tbgth2"

"Thisperson"

"Spike Trap"

"Player"

Why are these terrible Event Names or names for our boxes that we are creating? The first example makes no sense. While it is fresh on your mind, you might know what this Event will do; however, what about a month down the road when you start going back to find edit or fix bugs? What about if someone joins your team? While the second, third, and fourth examples are in fact cohesive names, they are too broad. What if you had fifty "Thisperson" on a map or 50 spike traps on a map? Which one are you referencing?

Good Name Examples:

"Lady Marie"

"Sir Edgar"

"Lady Marie Shot"

"Spike Trap 1"

"Spike Trap 2"

"Player Healing Point"

Looking at these examples, we see which person we are refencing. Not only that we are able to distinguish each Event apart, and get a general idea about what the Event will do. It makes no sense to name one of our boxes, "Toy Box," then cram it full of glass jars.

This is important. Because getting in the practice of properly naming Events and commenting Events will make you a better Game Developer and save you major headache later down the road.

I recommend keeping a notebook next to you or a word document where you write down important Events and characters. This practice will keep you organized, especially once we start going into Switches and Variables.

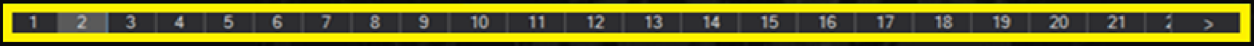
Global Event

This is important… all Events are Player Events. What this means, is when a player activates an Event, no one else on the map will see what is happening. So, in example, when a player opens a door by pressing a “Button Event,” only that player will see the door open. No other player will see it. By checking the Global Event box, this bypasses this rule and makes it affect everyone on the map.

Let’s take a closer look:

We have two players on a map: Cecil and Sara. Cecil has an iron key in his inventory and opens the door by a “Door Event”. He sees that the door is open and he can now walk through it with no problem; however, Sara still sees the door closed and locked. In fact, it looks like Cecil just walked through a closed door! By checking Global… ***BAM****. The door is opened for all players on the map when Cecil uses his key, so now Cecil and Sara both can walk through the door.*

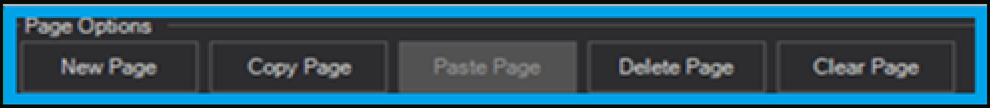
Page Number



Page Numbers: 1+

Remember the rule that every new box (Event) we create is empty? This is another rule: every new box (Event) starts with 1 Page within it. Think of each page as a divider that you may place inside our box to separate things. So, not only does our box have toys in it, but we divided the toys into two or more categories: toys with wings, toys with cars, toys with hats, or any other category we want to create. That way we keep everything separated. An Event with 1 Page in it has no dividers within it, an Event with 2 pages has two dividers, an Event with 3 Pages has three dividers, and so on. This is crucial when we get into Event Conditions. Refer to the advanced section: Event Pages and Conditions.

Page Options



New Page

Each time you press the "New Page" button, it adds a divider (page) to our box (Event). Create only the amount needed for the Event, and do not create one-hundred pages and jump between them randomly for fun. It will not work out well for us.

Copy & Paste Page

These buttons allow you to copy the entirety of a page to your clipboard. The Copy/Paste function will save you an enormous amount of time when using similar events on the same or different maps. Let's say we have a 10-page Event, with changing dialogue on different pages for Lady Marie. If we did ten pages by hand this is what it would look like:

(1) Set Character Sprite, Movement, conditions, and anything else we needed.

(2) Then we must create each Event Command and conditional branches all over again.

(3) We have to do this ten times for our Event, with increasing probability of errors.

We will never finish anything if we did it all by hand. That is where our Copy-Paste friend comes into play. We copy a page, then paste a new page. Our second, third, fourth, all the way to tenth page are all the same now and took less than 60 seconds to do it. Now we can just go through and change dialogue, conditions, and/or anything else we want on each individual page. This a considerably less amount of work and less chance of messing something up.

*Outside the Event, we can hold Ctrl + C and click the Event, then Ctrl+V and click a new part of the map to copy-paste the entire Event from location to location or map to map.

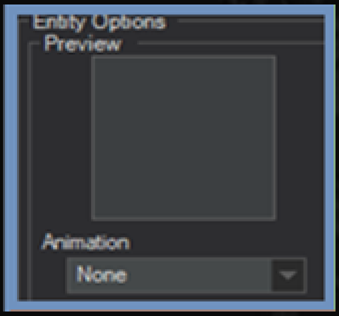
Delete Page

Oops! We made too many pages! No problem, just click the page number and hit Delete Page button. All done, but be careful to not delete the wrong page!

Clear Page

Oops! We really messed something up on this page. We want to keep the page's location within the Event, but we want everything cleared? This is the button for us; by clicking the Clear Page button, all the data within the selected page is blanked out as if it were a new page!

Entity Options



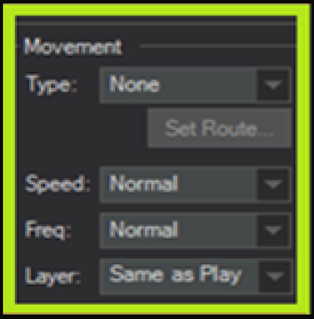
Preview

This is where we set the decoration of our box (The graphic the player sees on the map). Without placing a graphic, the Event is invisible and the player is unable to see it. Essentially, we painted our box with invisible paint. We want to set graphics that match what our box is: a person graphic our "Person Box," or a treasure chest graphic for our "Treasure Chest Box." The choices are numerous and is one of the best parts of designing your game.

Animation

Whenever the Event is interacted with, a chosen animation plays on our Event. This means we can set our box to also become a music box. it could play a smiley face with a sound of a giggle.

Entity Options: Movement


Movement
Type: None
Set Route...
Speed: Normal
Freq: Normal
Layer: Same as Play

Type:

None

Our Event will not move, unless it is forced to move via Event Commands.

Random

The Event will move Randomly on the map. This is often used for a village NPC or monster.

Move Route

This Movement Type is the one you will be using the most within mobile Events. We can set it approach or flee from the player. We can set speed, jumps, and a wide variety of other types of Movements. This is a bit trickier, but is more fun.

Speed

After we set a Movement Type, we need to set the Speed of the Event. When the Event moves, how fast is it moving? Does it saunter, shamble, dash, or frolic?

Frequency

After we set a Movement Type, we need to set the Frequency of the Event. The Event will follow the assigned movement type; however, the frequency affects how often the

Event will move. So, slower speeds make the Event move rarely, while a higher speed makes it move more often.

Layer

The chosen layer is crucial for our Event. Should we choose an incompatible layer with what we want for our box, the graphics will look skewed to the player. Think of it as being in front of the box, above the box or below the box. For most cases, we will be using Same as Player, but let us take a closer look at the three choices:

Same as Player: The Event is layered to cause no overlap with the player.

Below Player: The player always looks like they are standing on the event.

Above Player: The Event is always above the player (on their head).

Now what does this mean and how can we use it? Let us look at three different scenarios.

In the first scenario,

We are creating Lady Marie as a village NPC; her goal is to simply walk around randomly and talk when a player activates her. In this instance, we will use Same as Player layering. This means Lady Marie is on the same plane of existence as us and we will bump into each other should we walk into each other.

In the second scenario,

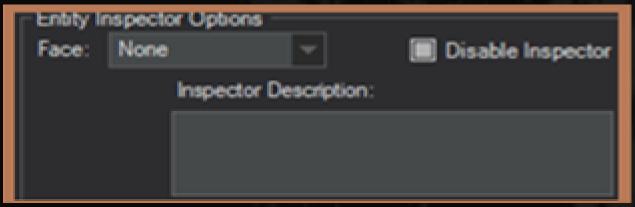
We want to make a spike trap. By using the Below Player layer, the spikes appear to be below the player with them standing on top of it. We are stepping on the trap, so it cannot be bumping into us or over our head.

In the third scenario,

We are creating a bird that flies overhead; thus, we need to use the Above Player layer. Now when the bird occupies the same space as us, it appears over our character because it is flying. If we bumped into each other, it would mean we both are flying! Then if it was below us, it would mean we were flying and they were crawling!

Looking at the different types of layering, we can see how important it is to apply the correct layer or it will cause the game to look "buggy." It also allows the developer to experiment with the layers and do creative things with them.

Entity Inspector Options



Face

This is where we put a picture over the keyhole, also known as assigning the Event a face when activated by the player. The player selecting this Event can see the face (picture of lock) when activated. What this means is by selecting this Event (inspecting), the face will show up with its description beside it (if applicable).

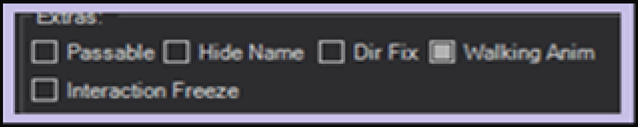
Disable Inspector

The Event cannot be inspected; thus, the face is never shown on this event.

Inspector Description

Selecting the Event (Inspecting), will make a window popup with this description in it.

Extras



☐ Passable ☐ Hide Name ☐ Dir Fix ▣ Walking Anim
☐ Interaction Freeze

Passable

Ow! You just walked into your box and it hurt. Likewise, in your game the player cannot walk through the box (Event). When enable the "Passable" button, we effectively make our Chest turn into a ghost chest! Spooky! We can now walk through it without ever knowing it is there. This is usually used in video games when making a closed door open.

Hide Name

Remember when we discussed the importance of naming Events? This box hides the Event Name from being seen by anyone while in the game.

Direction Fix

When you activate or talk to an Event, the Event turns and faces the player from the direction they were activated from. Checking this box makes the Event to always face the initial direction we selected for it.

Walking Animation

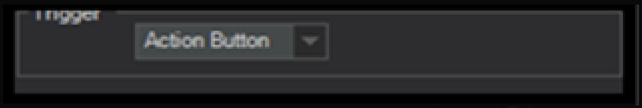
Checking this box allows the sprite to play through its sprite sheet to create a walking animation. This is necessary for a NPC walking around; however, we want this box checked if we have several different sprites on the same sheet, as it will just change the graphic between them.

I.E: Having several treasure chests on 1 sheet.

Interaction Freeze

This freezes the Event in place when activated. Without having this box checked, the Event will continue to move while it runs. This box is necessary if you want NPCs to stand in place to talk to us, else they just wander away while chat boxes still appear.

Triggers



[Trigger]

There are three ways for the Event to be activated.

Action Button

The player must use their action button on the Event to activate it. This is a familiar concept, "push the button to talk to the NPC."

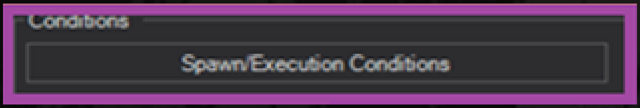
Player Touch

When the player touches the Event, the Event activates.

Auto-run

This Event activates automatically when the player loads on the map.

Conditions



Spawn/Execution Conditions

Okay…. We are going to have to come back to this. This is some advanced stuff we will be discussing here, and we have not gone through everything yet to tackle this topic. We will go into Event Commands first, then tackle Switches, Variables, Conditional Branches, and Conditions after we gone through all the Event Commands.

Event Commands

Event Commands

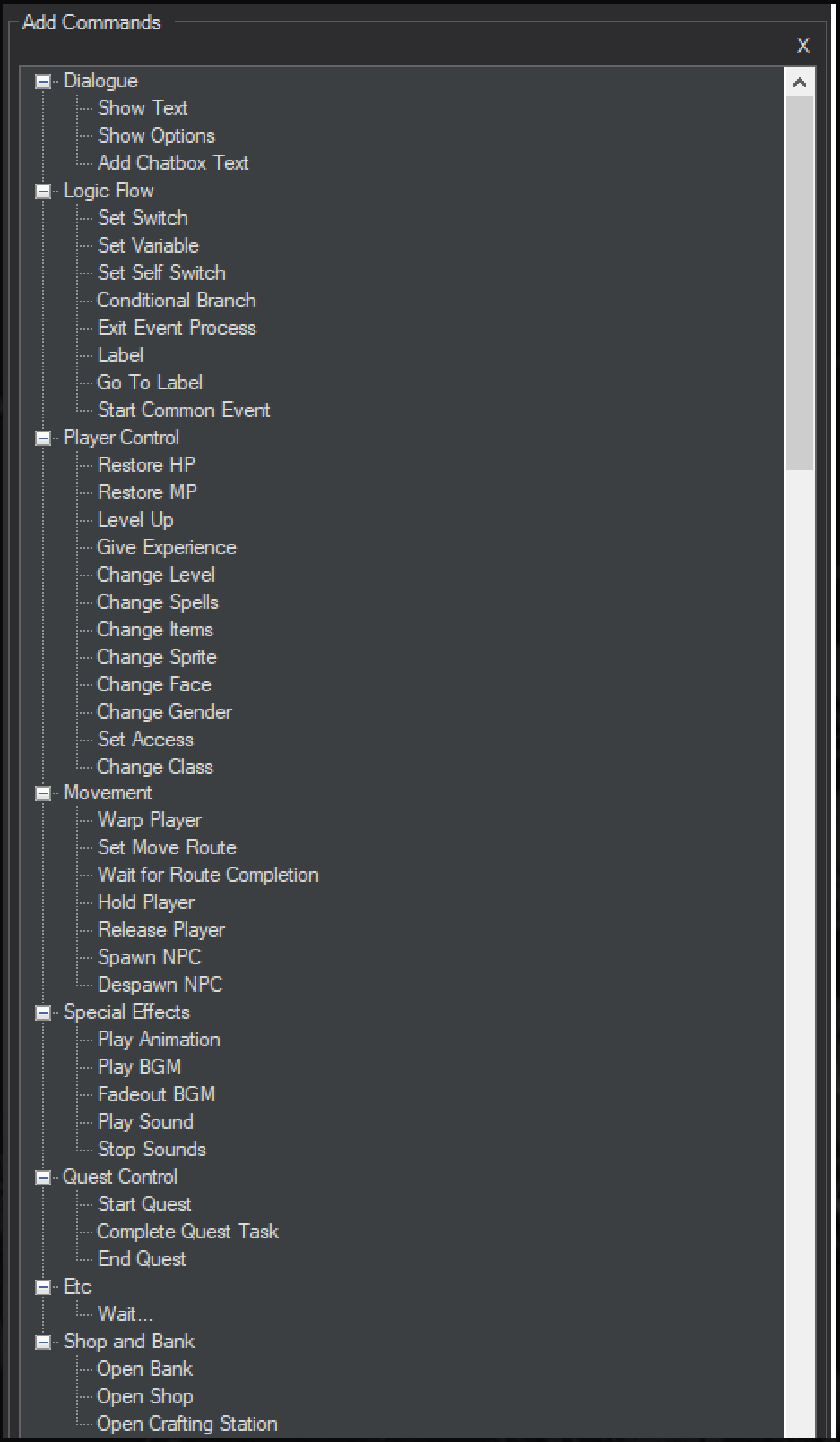
Event Commands can be simple or complex as the developer needs them to be. Before we can discuss Complex Event Commands, we need to understand the Simple Event Commands and how they work. So, following our box and clay analogy, let us look at the Event Commands.

Event Commands are the blocks of clay that we are taking from our boxes and molding into different objects. Essentially, each clay object we make is an Event Command that is placed in our box. When we activate the Event by a Trigger, what we are doing is picking up each clay object in our box one at a time. This is always done by picking up the first one, then we play with it based on what it is, then we place it behind the very last clay object. We repeat this process until we have gone through all the clay objects.

What this means in game terms: each Event Command is ran in descending order from the first Event Command until the last when the Event is triggered.

At its basest of levels, an Event operates in the following way: The *Event is Activated*, the *Event runs every Event Command*, and finally the *Event Ends.* Before we can get into Complex Event Commands, we need to go through each Event Command. Afterwards we are going to go into Complex Events, Conditional Branches, Switches, and Variables. Do not worry, we will get through this. Now for the easy part, Event Commands:

Command Masterlist

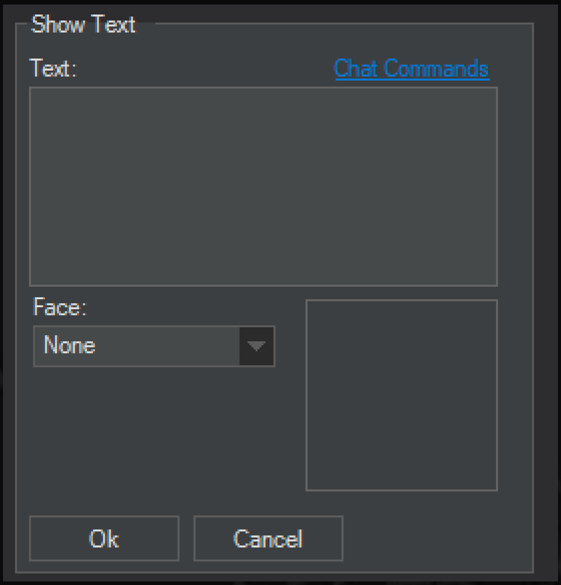

Add Commands

X

- Dialogue
  - Show Text
  - Show Options
  - Add Chatbox Text
- Logic Flow
  - Set Switch
  - Set Variable
  - Set Self Switch
  - Conditional Branch
  - Exit Event Process
  - Label
  - Go To Label
  - Start Common Event
- Player Control
  - Restore HP
  - Restore MP
  - Level Up
  - Give Experience
  - Change Level
  - Change Spells
  - Change Items
  - Change Sprite
  - Change Face
  - Change Gender
  - Set Access
  - Change Class
- Movement
  - Warp Player
  - Set Move Route
  - Wait for Route Completion
  - Hold Player
  - Release Player
  - Spawn NPC
  - Despawn NPC
- Special Effects
  - Play Animation
  - Play BGM
  - Fadeout BGM
  - Play Sound
  - Stop Sounds
- Quest Control
  - Start Quest
  - Complete Quest Task
  - End Quest
- Etc
  - Wait...
- Shop and Bank
  - Open Bank
  - Open Shop
  - Open Crafting Station

Dialogue

Show Text



Text

This is also known as a Dialogue Box. We can put text here for NPC's to say or add descriptions

to Events we click.

Face

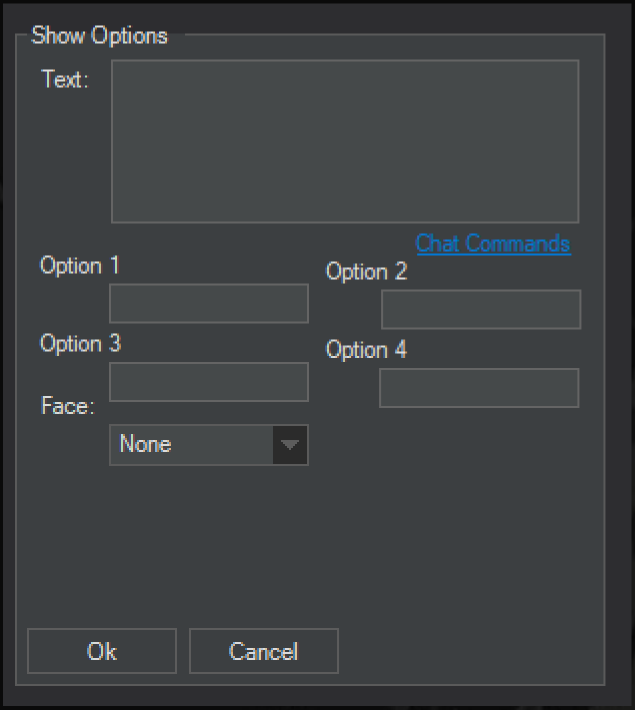
This is the Face that appears in conjunction with the Dialogue box.

Chat Commands (Taken directly from Jc's post)

- \pn -- shows the players name that triggered the event
- \en -- shows the name of the event that is activated
- \onlinecount -- shows the number of players that are online
- \onlinelist -- lists all the players that are online
- \param -- takes the string after a /command. Example: /welcome kibbelz, will return "kibbelz" in the text display. Available from beta 4.0 onwards.
- \hour -- displays the hour component of the games time (12 hour format)
- \24hour -- displays the hour component of the games time (24 hour format)
- \minute -- displays the minute component of the games time
- \second -- displays the second component of the games time
- \period -- displays whether or not it is morning/night in game (AM or PM)
- \pv # -- displays the value of a player variable (replace # with the variable id shown in the switch & variable editor)

- \ps # -- displays the value of a player switch (replace # with the switch id shown in the switch & variable editor)
- \gs # -- displays the value of a global switch (replace # with the switch id shown in the switch & variable editor)
- \gv # -- displays the value of a global variable (replace # with the variable id shown in the switch & variable editor)

Show Options



Text

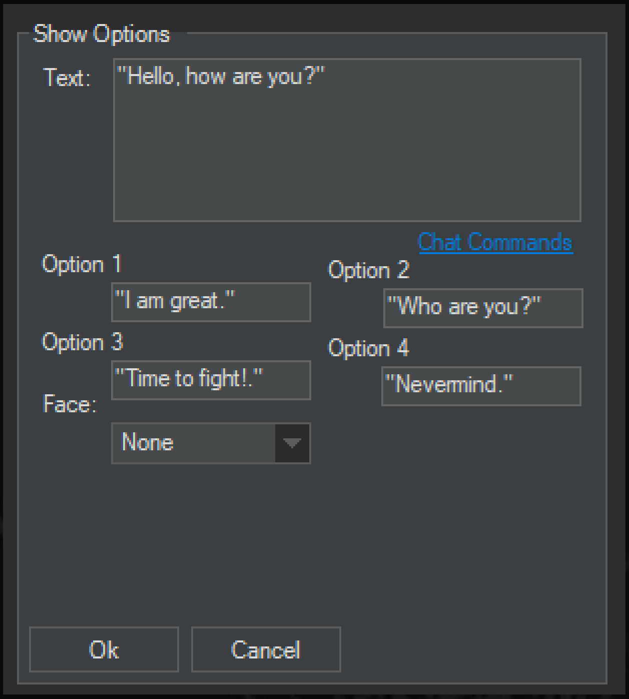
This is also a known as a dialogue box. Unlike the Show Text, this dialogue box allows us to add options or responses for the player to choose from. That means choices!

Option 1 – 4

We can create up to 4 responses for the player to choose from. For each option, we create, we must add Events to the Event Tree for each option. These are called branching Events.

WOAH, Woah! Let us back that swanship up. What does that even mean?

No stressing, we got this. First, we need to fill out the options:



Text: "Hello, how are you?"

Option 1: "I am great."

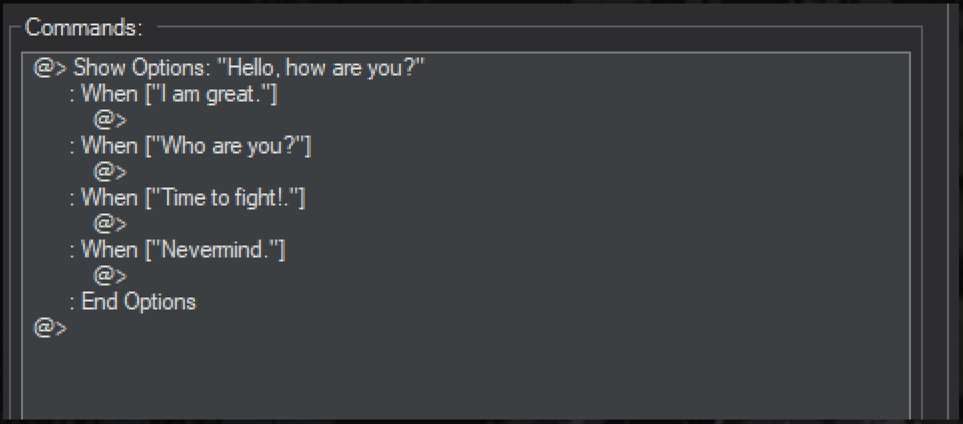
Option 2: "Who are you?"

Option 3: "Time to fight!."

Option 4: "Nevermind."

Face: None

Okay, so we got it filled out and this is what our screen looks like now. This looks a bit complicated, but do not worry we just need to break it down to understand it.



```
@> Show Options: "Hello, how are you?"
     : When ["I am great."]
        @>
     : When ["Who are you?"]
        @>
     : When ["Time to fight!."]
        @>
     : When ["Nevermind."]
        @>
     : End Options
@>
```

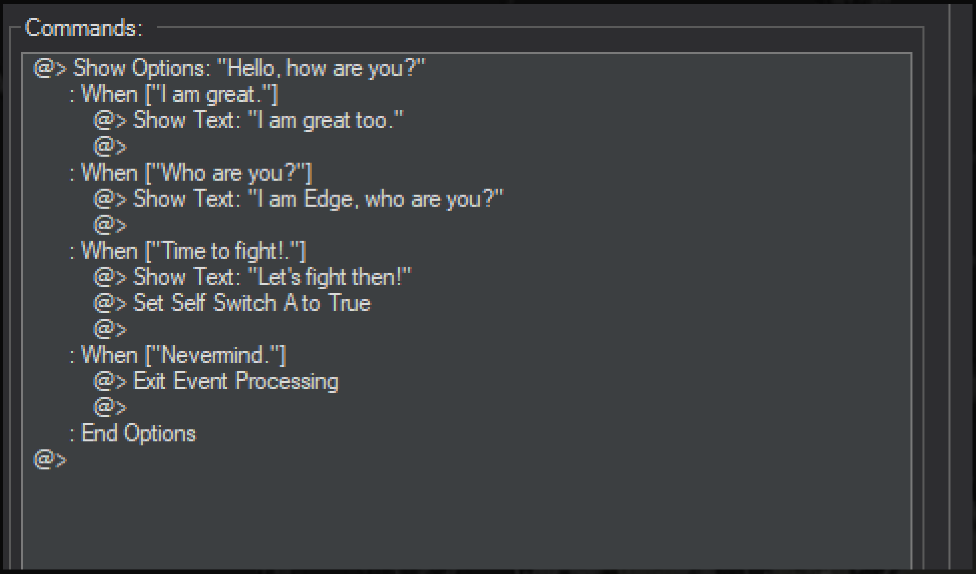
There are two crucial things we need to understand here:

When: These are our options that we created. We created 4, so there are 4. If we did 3, there would be 3 and so on. Basically, it is saying, “When the player selects this option, do this.”

@: This symbol means that all the Events with this Symbol are connect to the choice (When) directly above it.

Is it starting to make sense? Let's take it further and apply Events or the answers to our questions.

Here are a few examples of what some answers could look like:



```
Commands:
@> Show Options: "Hello, how are you?"
    : When ["I am great."]
      @> Show Text: "I am great too."
      @>
    : When ["Who are you?"]
      @> Show Text: "I am Edge, who are you?"
      @>
    : When ["Time to fight!."]
      @> Show Text: "Let's fight then!"
      @> Set Self Switch A to True
      @>
    : When ["Nevermind."]
      @> Exit Event Processing
      @>
    : End Options
@>
```

So, taking what we learned let us look how this would play in the game.

The NPC will say, "Hello, how are you?"

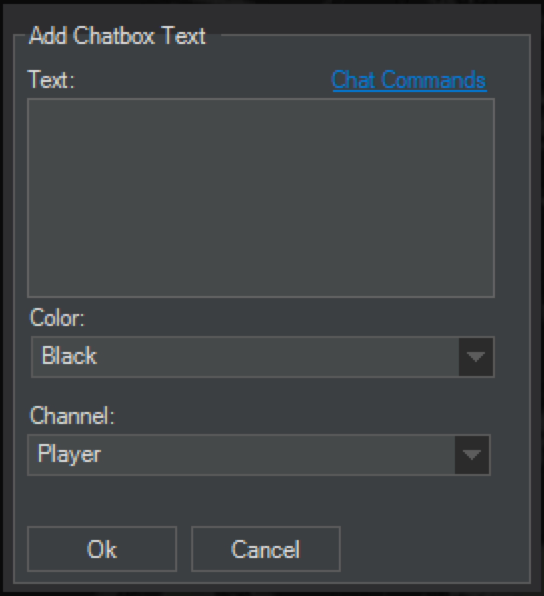
1. The player is given 4 responses to choose from.
2. Selecting, "I am great." Will cause the Event to say, "I am great too."
3. Selecting, "Who are you?" will cause the Event to say, "I am Edge, who are you?"
4. Selecting, "Time to fight!" will cause the Event to say "Let's fight then!" **and** will set the Self-Switch to True.
5. Selecting, "Nevermind." Will cause the Event to exit.

6. The @ at the end allows you to place Events to run after any of the choices are selected. If there are no Events the Event ends.

Selecting a choice runs the created Event; however, we created two for “Time to Fight!”, so both of those Event Commands run. We can create as few or many as we want for each choice.

Add Chatbox Text

Chatbox text works differently than Show Text, as this does not make a window pop up. Instead it puts the text directly in the chatbox.


Add Chatbox Text

Text:

Chat Commands

Color: Black

Channel: Player

Color

We can set the color of the text; however, we need to be careful to make sure we do not make the color too dark or bright against the chatbox background as it may become unreadable.

Channel

We can set the chatbox text to be sent to the player only, everyone on the current map, or to every player on the server.

Player Control

Restore HP

This Event Command instantly and fully heals the player’s Health Pool.

Restore MP

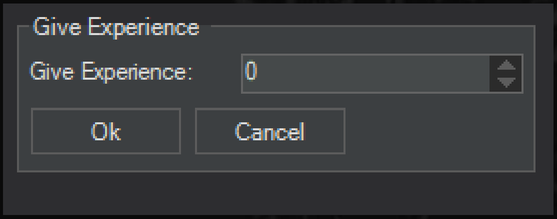
This Event Command instantly and fully heals the player’s Mana Pool.

Level Up

This Event Command instantly fills the player’s Experience bar until they level up and restarts their experience back at 0.

Give Experience

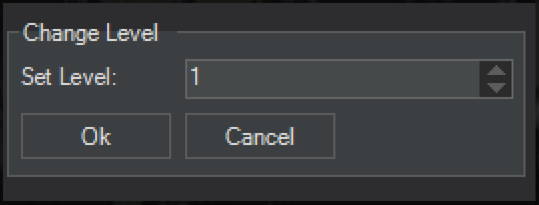
This Event Command grants the player a set amount of experience, that you can set.


Give Experience

Give Experience: 0

Change Level

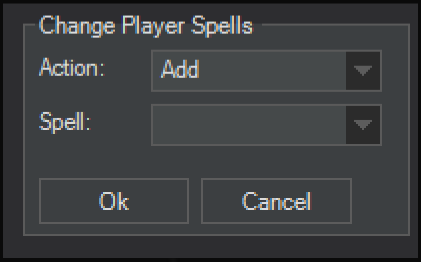
This Event Command changes the Player’s Level to specific level. Previous Attribute Points that are distributed are not lost; furthermore, leveling up grants you additional Attribute Points to spend.


Change Level

Set Level: 1

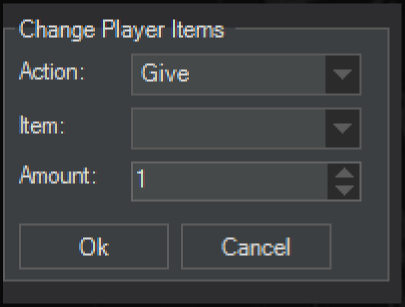
Change Spells

This Event Commands allows you to grant or remove a chosen spell to the player. It is important to note that the Spell Icon, on the hotbar, will need to be manually added or removed.


Change Player Spells
Action: Add
Spell:

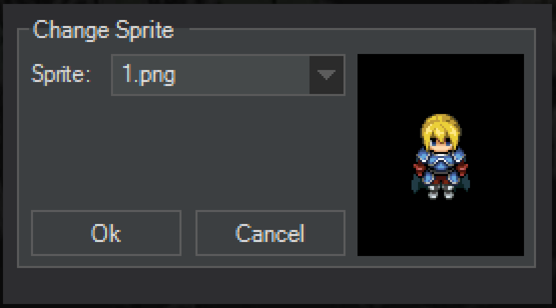
Change Player Items

This Event Command allows you to add or subtract a finite number of items from the Player's Inventory. It is important to note this does not affect equipped items.


Change Player Items
Action: Give
Item:
Amount: 1

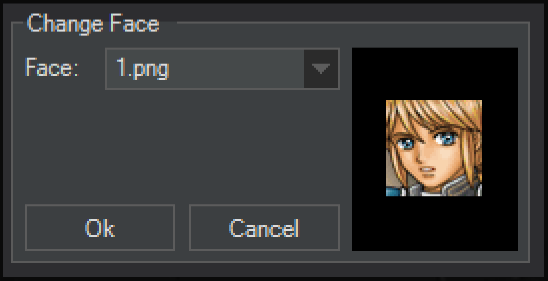
Change Sprite (Player)

This Event Command allows you to permanently change the Player's character sprite to a new one.


Change Sprite
Sprite: 1.png

Change Face (Player)

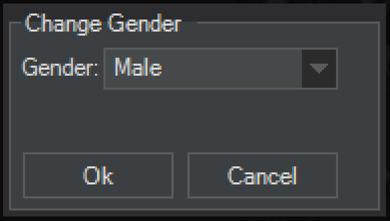
This Event Command allows you to permanently change the Player’s face sprite to a new one.



Face: 1.png

Change Gender (Player)

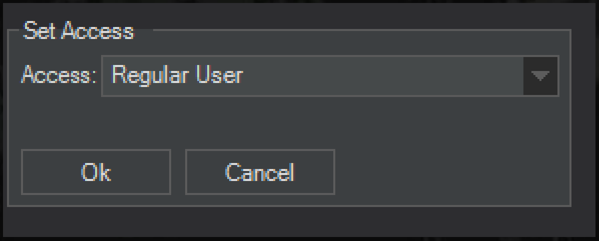
This Event Command allows you to change between two of Intersect’s default genders; however, with the source code we can add as many choices as we want. This mainly affects conditions for spells, items, and paperdoll rendering.



Gender: Male

Set Access (Player)

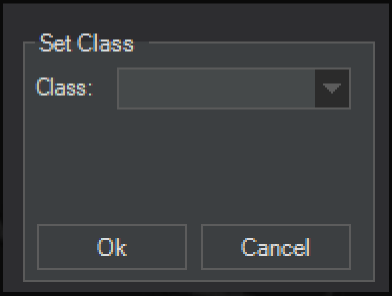
This Event Command allows you to make someone a Moderator or Administrator for your game. You can also remove their privileges and revert them back to regular users.



Access: Regular User

Set Class (Player)

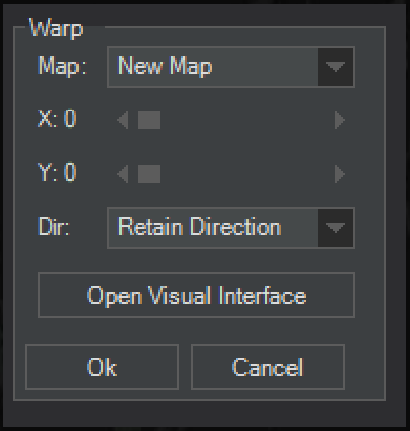
This Event Command allows you to change the player's class to a new one. It is important to note that class restrictions and conditions can change as well. In short, migrating from an old class to a new one might render certain equipment or spells unusable.



Movement

Warp Player

This Event Command allows the Event to instantly warp the player into a new map. Additionally, we can set the player's direction, X/Y position on the map, or use the interface to select a spot.



Move Route

There are two sections to the Move Route Event Command that we need to understand. The first one is setting the Move Route of the player; the second is the Move Route for an Event.

On both Routes, there are two check boxes, *Ignore if Blocked* and *Repeat Route*, located in bottom left hand corner. If something prevents the Event or Player from moving it can “freeze” the game for the player. The *Ignore if Blocked* condition helps prevent that from happening.

Ignore if Blocked

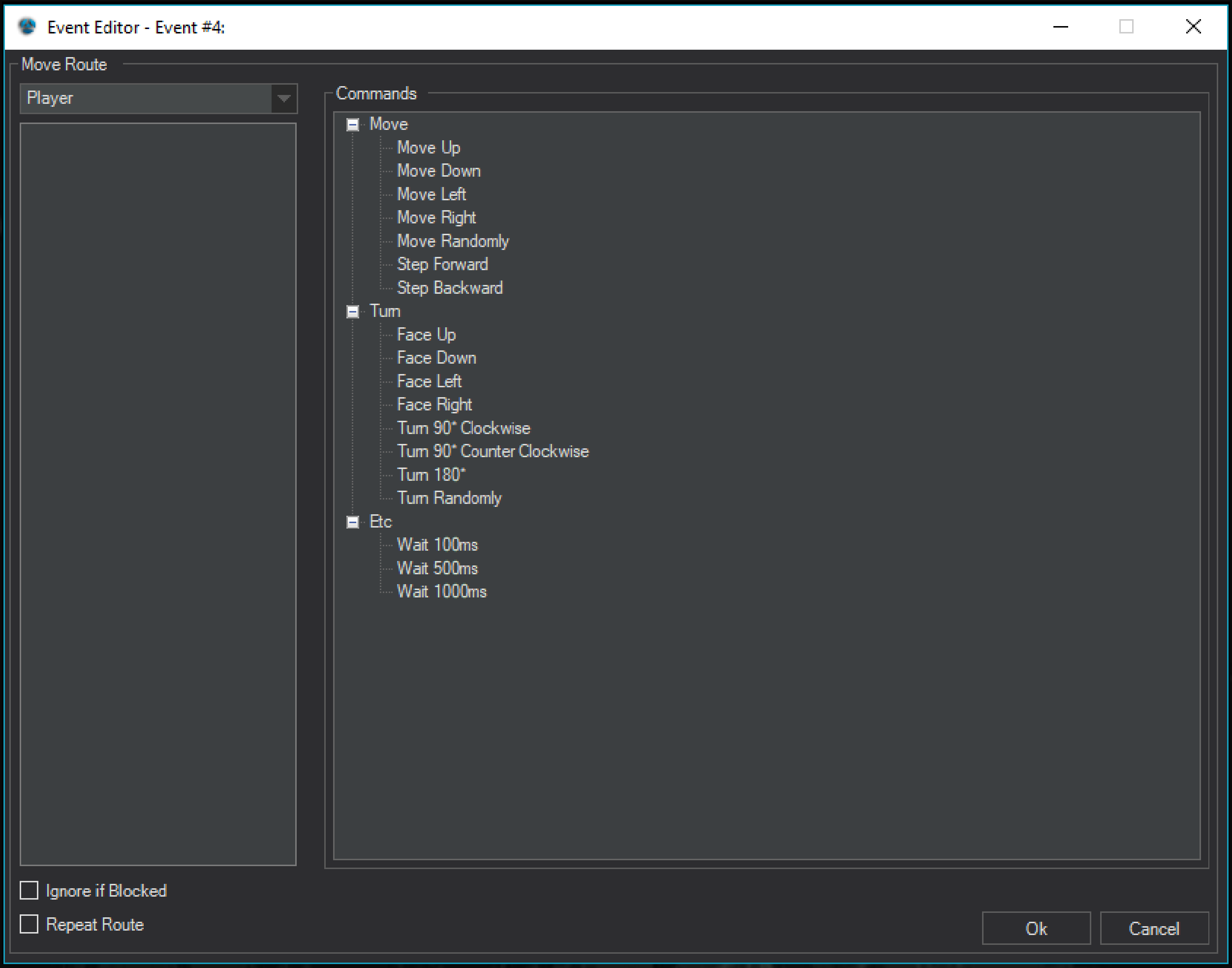
The Movement Route is canceled when it becomes blocked by running into an obstacle that prevents movement of its route.

Repeat Route

The Player or Event will do the entire Move Route and repeat it from the beginning each time it is cleared.

Player Move Route

We can now issue Commands that force the player to move in a desired path. This is great for story mode cinematics or Events that trigger movement (I.E: jumping on stones over a river). The Commands are pretty self-explanatory.


Event Editor - Event #4:

Move Route

Player

Commands

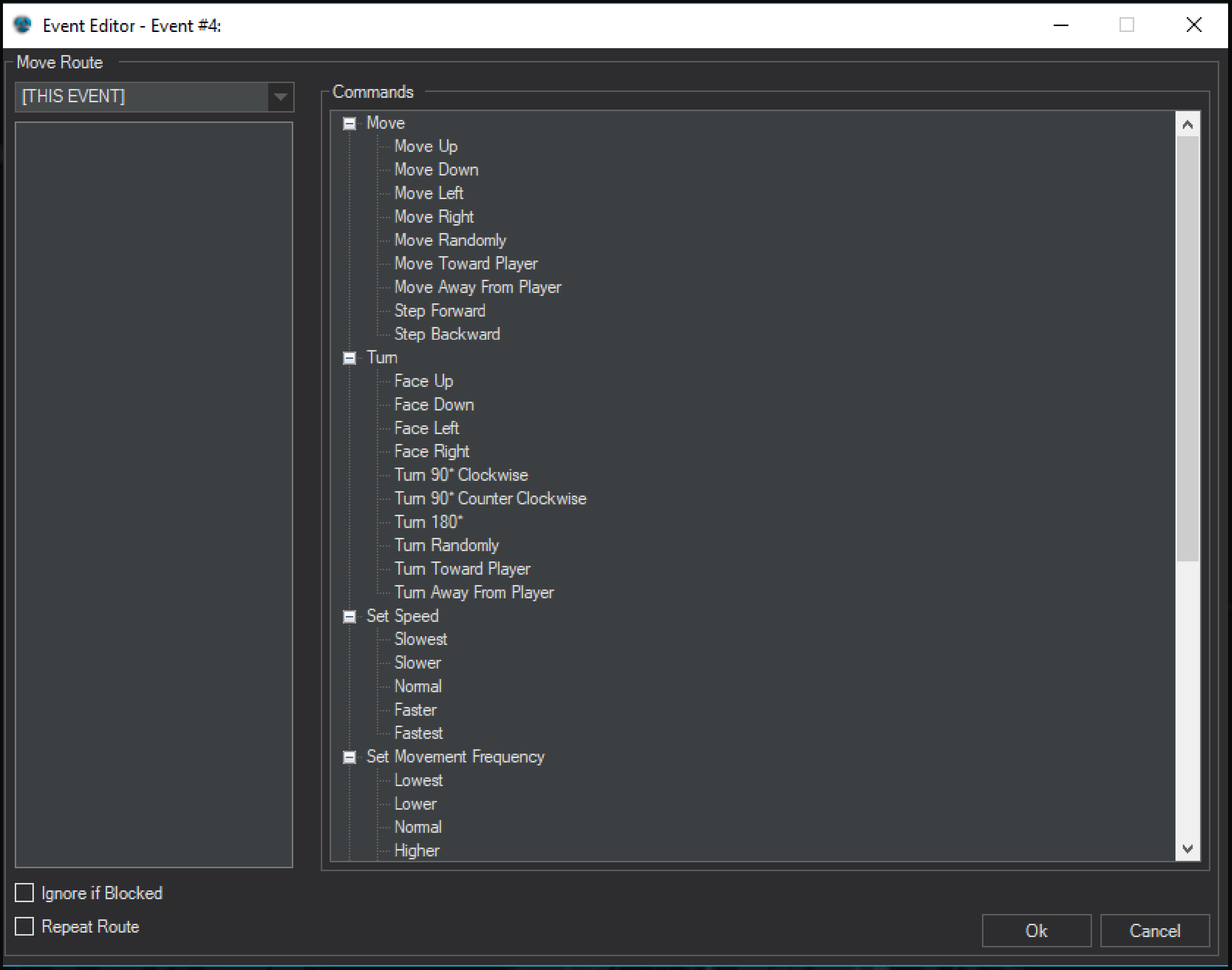
- Move
  - Move Up
  - Move Down
  - Move Left
  - Move Right
  - Move Randomly
  - Step Forward
  - Step Backward
- Turn
  - Face Up
  - Face Down
  - Face Left
  - Face Right
  - Turn 90° Clockwise
  - Turn 90° Counter Clockwise
  - Turn 180°
  - Turn Randomly
- Etc
  - Wait 100ms
  - Wait 500ms
  - Wait 1000ms

☐ Ignore if Blocked

☐ Repeat Route

Event Move Route

Looking at the Event Move Route, we can see it is more advanced than the Player Move Route. In fact, we have two pages of Commands to go through. Most of the commands are self-explanatory as well as commands to override the base Event's Movement we set earlier. There are a few we need to take a closer look at.

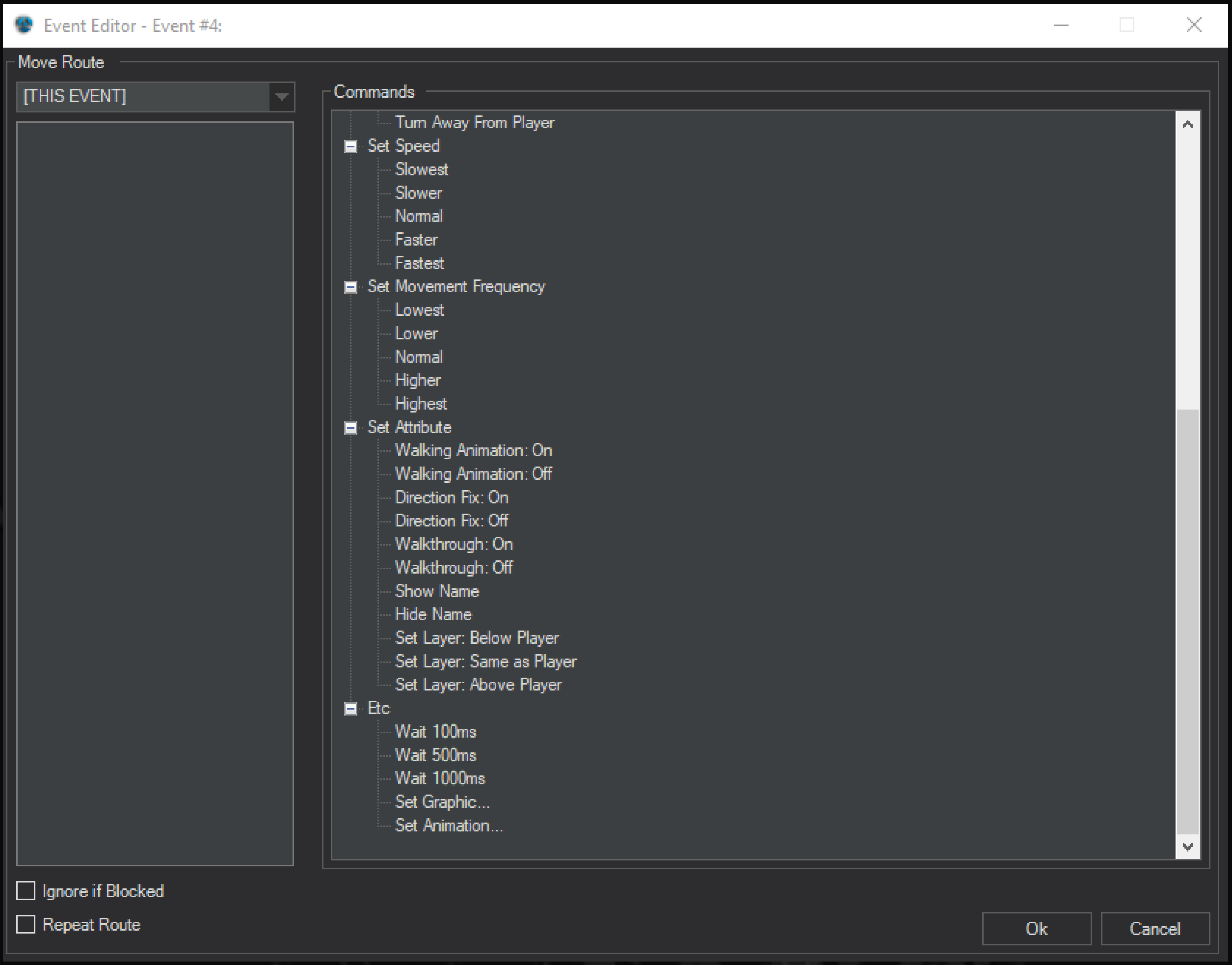

Event Editor - Event #4:

Move Route

[THIS EVENT]

Commands

- Move
  - Move Up
  - Move Down
  - Move Left
  - Move Right
  - Move Randomly
  - Move Toward Player
  - Move Away From Player
  - Step Forward
  - Step Backward
- Turn
  - Face Up
  - Face Down
  - Face Left
  - Face Right
  - Turn 90° Clockwise
  - Turn 90° Counter Clockwise
  - Turn 180°
  - Turn Randomly
  - Turn Toward Player
  - Turn Away From Player
- Set Speed
  - Slowest
  - Slower
  - Normal
  - Faster
  - Fastest
- Set Movement Frequency
  - Lowest
  - Lower
  - Normal
  - Higher



```
    Turn Away From Player
Set Speed
    Slowest
    Slower
    Normal
    Faster
    Fastest
Set Movement Frequency
    Lowest
    Lower
    Normal
    Higher
    Highest
Set Attribute
    Walking Animation: On
    Walking Animation: Off
    Direction Fix: On
    Direction Fix: Off
    Walkthrough: On
    Walkthrough: Off
    Show Name
    Hide Name
    Set Layer: Below Player
    Set Layer: Same as Player
    Set Layer: Above Player
Etc
    Wait 100ms
    Wait 500ms
    Wait 1000ms
    Set Graphic...
    Set Animation...
```

☐ Ignore if Blocked
☐ Repeat Route

Set Graphic

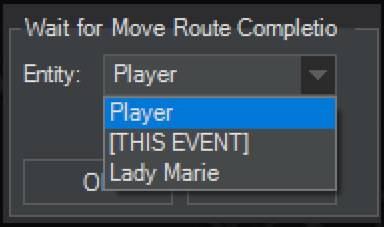
This allows us to change the base Event's graphic to a new graphic.

Set Animation

We can have an animation play over the Event. This is great in helping set up story Events or effects for objects.

Wait For Route Completion

This handy command allows you to stop all triggers within the Event to stop firing, until a chosen entity finish moving (Player, This Event, or another Event). Once the entity finishes its move, the trigger continues to proceed.



Hold Player

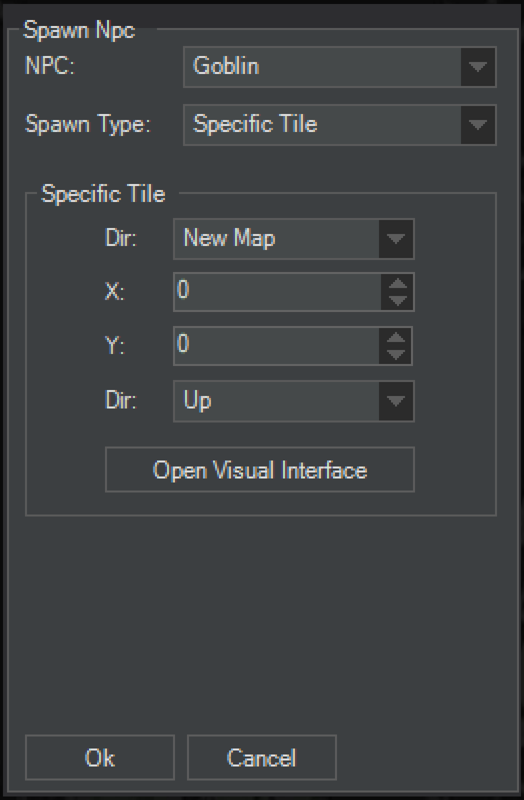
This command is essential in cutscenes because it prevents the player from running around during conversations or other cinematic Events.

Release Player

This command is to be used to release the **Hold Player** command.

Spawn NPC

This Event Command is essential for spawning in creatures after cinematics, or when only have the monster spawn after a quest has been started. This Event Command allows you to choose a NPC (Monster) and place it on a specific tile, around an Event or Player. While you can use X and Y coordinates, it is sometimes more efficient to just **Open Visual Interface** to click where you want the NPC to spawn at.



Spawn Npc

NPC: Goblin

Spawn Type: Specific Tile

Specific Tile

Dir: New Map

X: 0

Y: 0

Dir: Up

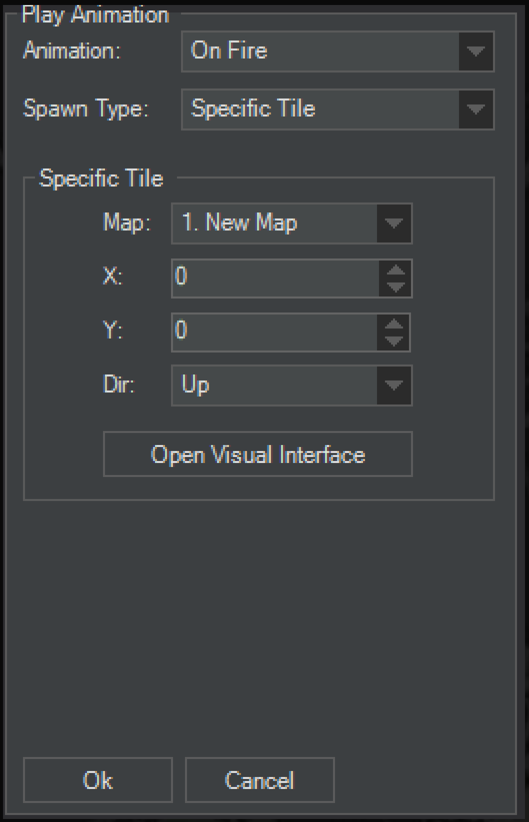
Despawn NPC

This Event Command despawns all NPCS on the map?

Special Effects

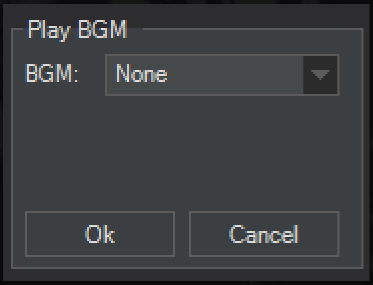
Play Animation

This Event Command allows you to play a chosen Animation on the player, Event, or specific tile. Once again you see we can choose a X, Y axis and Map; however, using the **Open Visual** Interface is usually used.


Play Animation
Animation: On Fire
Spawn Type: Specific Tile
Specific Tile
Map: 1. New Map
X: 0
Y: 0
Dir: Up

Play BGM

Changes the music currently playing to another one. It is important Intersect only supports .mp3 files for music.

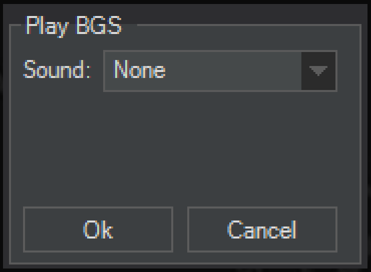

Play BGM
BGM: None

Fadeout BGM

The current music playing is slowly faded out into silence. This could be used cinematically to change the mood or slowly shift into another song without an abrupt cutoff.

Play Sound (BGS)

This plays a sound effect with the player at the center of it. It is important to note that Intersect only plays .wav files for sounds.


Play BGS
Sound: None

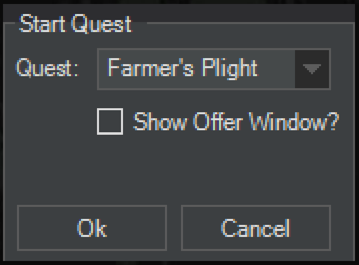
Stop Sounds

This stops all sound effects currently being played.

Quest Control

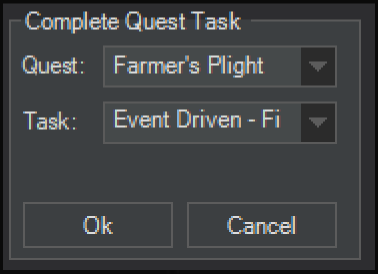
Start Quest

This Event command starts a quest for the player. You can also enable **Show Offer Window** to allow the player the choice of accepting the quest or not.


Start Quest
Quest: Farmer's Plight
☐ Show Offer Window?

Complete Quest Task

This Event Command allows you to supersede the original Quest parameters and complete tasks for a quest via the Event Command, rather than its original intention of the quest.



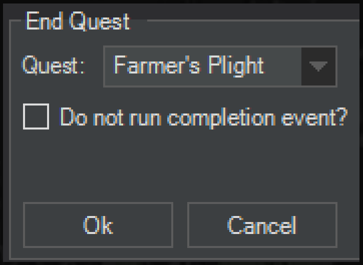
Complete Quest Task

Quest: Farmer's Plight

Task: Event Driven - Fi

End Quest

This Event Command allows you to end the quest prematurely. This is usually done because the player has done something that the game creator deemed to "break" the quest. Normally the quest ends and the player gains no reward; however, you can also set it to end if a player has done something special to end it early. In this case you can check the box to have it still reward the player because they have found a secret way of completing it or something else.



End Quest

Quest: Farmer's Plight

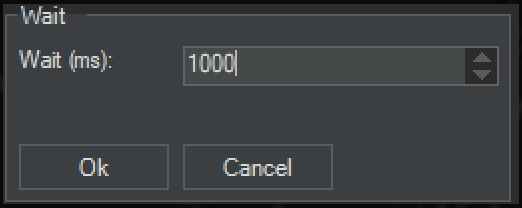
☐ Do not run completion event?

ETC

Wait

This Event Command is used to add “Pauses” to an Event. The Event Command will stop “to breathe” for how many milliseconds you order it to. After its “breath,” it will continue running the rest of the commands. Every 1,000 milliseconds, equals 1 second in game time.

It is important to not go below 1,000



Wait

Wait (ms): 1000

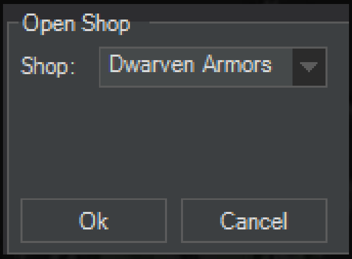
Shop and Bank

Open Bank

This command opens the player’s bank vault.

Open Shop

This command allows you to open a specific shop (that you created) for the player.

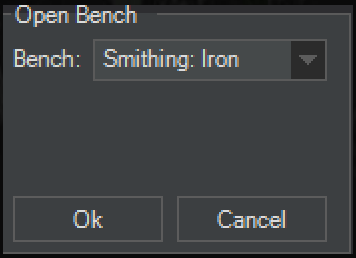


Open Shop

Shop: Dwarven Armors

Open Crafting Bench

This command allows you to open a specific crafting bench (that you created) for the player.

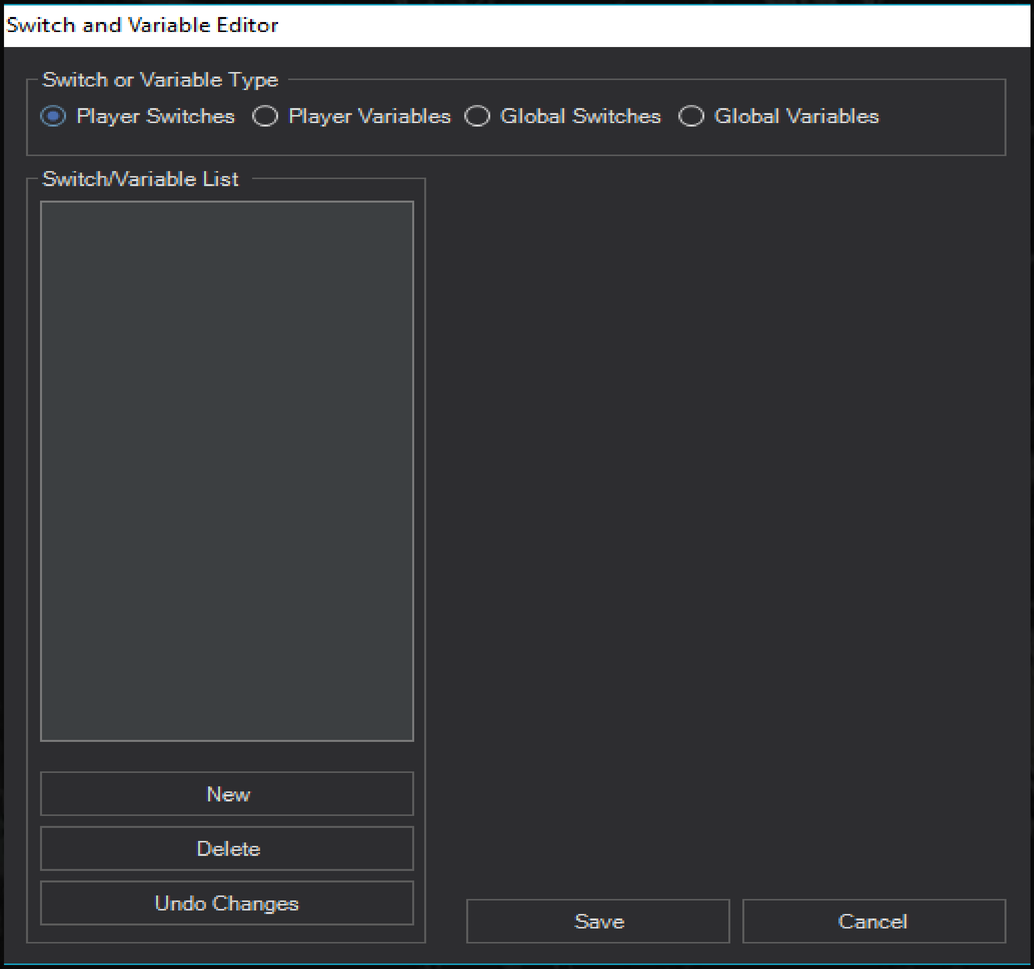


Open Bench

Bench: Smithing: Iron

Player Switches, Player Variables, Global Switches, and Global Variables! Oh, my!

It is time for the complicated stuff now. This might be completely new to you or just a refresher, but do not worry. We got this! We will just break it down piece by piece, as we have been doing this entire tutorial. This is where the fun begins as we become more familiar with the Event System, the more cool and fun stuff we can do to make our game even better! Trust me, once you learn how to use Conditional Branches, Variables, and Switches, your game development skills will evolve to a higher level of game creation.



Switches

What is a Player Switch?

Player Switches are light switches. This is the easiest way to think about Player Switches. A standard lightbulb can be turned on (True) and turned off (False) by the flicking of a light switch. A Player Switch operates in the same way, and by flicking it on (True) or off (False) can manipulate any number of Events in different ways. By turning the light Switch on/off, you are sending an electrical current to activate/deactivate one or more Events in your game that rely on the Switch being on or off.

What is a Self-Switch?

Self-Switches are the special kind of Switches we need to talk about. Generally speaking, Self-switches operate in the same way as Player Switches, except in two major ways. The first being Player Switches can affect other Events, no matter their location; however, Self-Switches can only affect Event Commands within its Event. Also, Self-Switches reset upon re-entering the map, while Player Switches remain toggled on or off until changed.

Why even use a Self-Switch?

There are two primary reasons to use a Self-Switches in your game. The first is is to reduce clutter in your game by not wasting resources on Player or Global Switches on Event Commands that only run within a specific Event. The second is to use Events that you want to have reset when you enter the map again*. An Event can only contain four Self Switches

Note: If you are familiar with Rpgmaker's Event System, Self-Switches do not reset on that program, but they do reset within Intersect.

What is a Player Variable?

A Player Variable is like an empty box. You place objects in both boxes and Variables to be used at a later date. Like a box, we should write something to let you know what's in the box. One such example is writing "Kitchen Stuff" on a box that has kitchen stuff in it. It is important that like all good boxes, we can always put more stuff in or take stuff out. What is also really cool is we can place boxes within boxes! This of course means we can place smaller boxes (Variables and Switches) into bigger boxes (Events). We will get more into that when we get into conditions a few paragraphs down.

Why use a Player Variable?

I have been waiting for you to ask! I can assure you once you start using Variables, just like Switches and Conditional branches, you will wonder how we ever lived without them! Let's say we are wanting to create a Romance System with different NPCs as options. We created all our dialogue and choices within our Events; however, how do we check if they love (or dare say hate!) our character? That is right: Variables.

How is a Variable different than a Switch?

This is an incredibly important question we must address. Variables are able to only store numerical values, while Switches can only store True or False values.

Wait, what?

Let's look at it like this: all Variables are containers for liquids. You may put different amounts of liquid within the Variable container; however, you still can only place liquid within the container (numbers). Cramming anything else into our containers (Variables) is bad and breaks the containers, so we don't do it.

As we discussed earlier, Switches are like light switches. They can only go on (True) or off (False). Putting liquid (numbers) into them is bad, so we don't do it.

This means:

Variables – Numbers Only!

Switches = On (True) or Off (False) Only!

What are Global Variables and Global Switches?

We talked about the term *Player* and *Global* meant earlier, right? It means the same thing here. Player Switches and Player Variables only affect the player themselves because those values are exclusive to each person and can different from person to person. *Global* means that it affects every player in the game! So, if we set a Global Variable to 2 and a Global Switch to True, that means for every player that Variable is 2 and that Switch is True! This is helpful for grand scale Events happening within the game or other global type quests or features. Use the Player ones for personal quests and dialogue.

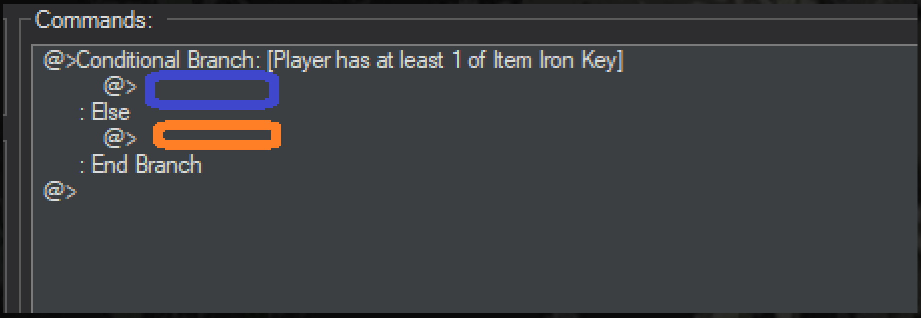
Conditional Branches

What is a Conditional Branch?

A Conditional Branch can be thought of as an indestructible lock we place our boxes to lock them and keep them safe. All our locked chests require a key to open it and only that one key can open it. A Conditional Branch checks to see if the conditions are met for it to open, it simply asks “Is this the key that opens me?” If the answer is yes it unlocks and runs the Event Commands. If the answer is no, it skips over the locked chest doing something else entirely.

Wait, I Do Not Understand

Let’s break it down then. My box (Event) now has a lock (Conditional Branch) now placed on it. When the Event activates it checks the Conditional Branch for an Iron Key. The locked on the box asks, “Do you have the Iron Key to open me?” If the answer is yes, the box opens and the Event Commands within the blue box are activated. If the answer is no, the box remains locked and the Event Commands in the orange box activate instead. It is important only the blue or orange color Event Commands will run, not both.



```
Commands:
@>Conditional Branch: [Player has at least 1 of Item Iron Key]
     @>
   : Else
     @>
   : End Branch
@>
```

*Note: The colored boxes are not actually in the engine, I just added them to highlight how they function.

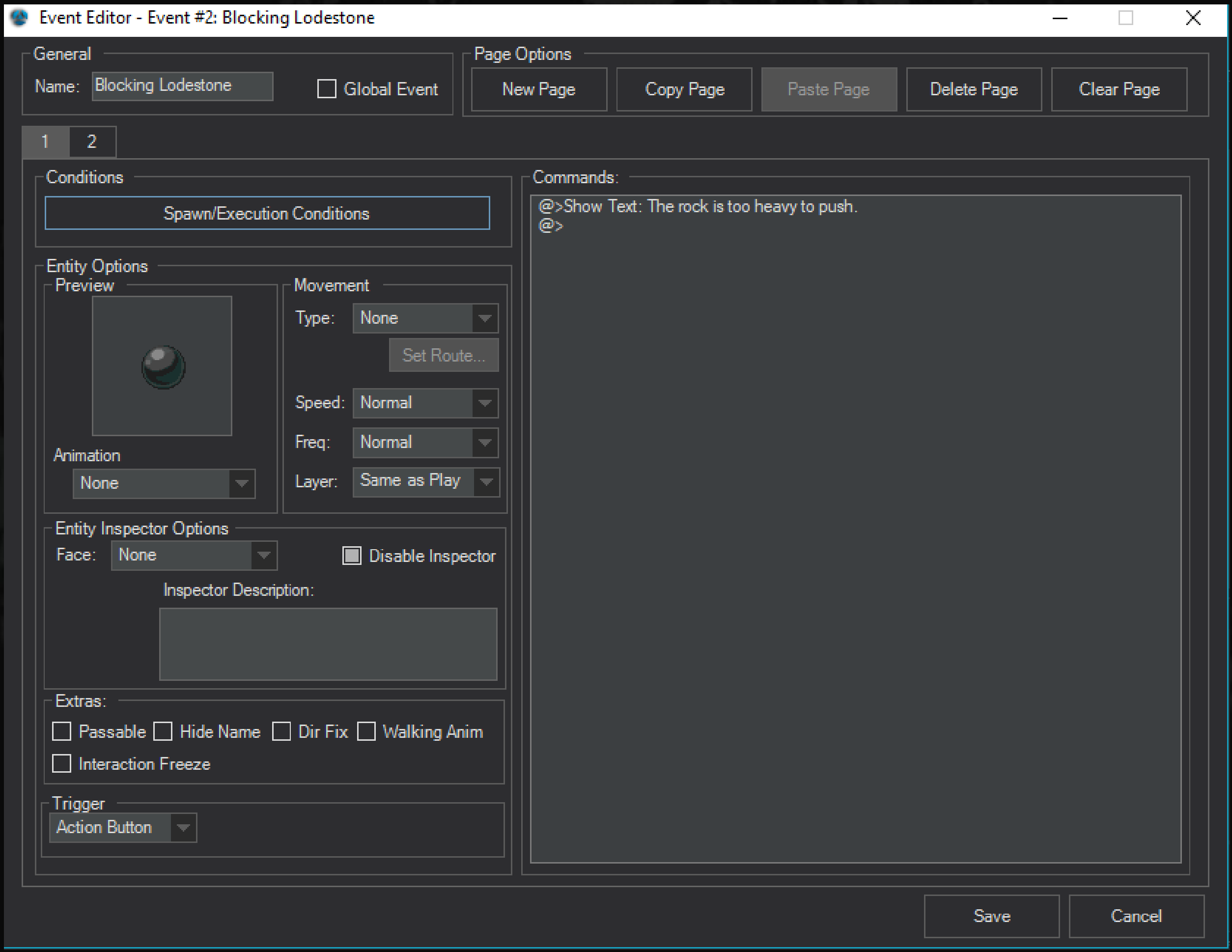
It is time… Let us bring it all together!

Now we have looked at everything, it is time to put it all together to create Complex Events. In most instances we will want an Event to have more than 1 page, because it will be doing multiple things and as a Game Developer we will need to learn to work with this. This may look a bit complicated at first, but all we are doing is applying the concepts we have discussed through the entire document. We will get through this. The most efficient way is to jump right in with an example. We will start with something simple: a movable rock.

Note: We will look at two different, but effective ways to use the box, lock, and key together.

Multiple Page Example:

So, we first set up our Event. It is pretty straightforward, you click the rock and it activates Page 1 and tells us the boulder is too heavy to push. We want the boulder to be moved only by a fighter; thus, we create a second page!


Event Editor - Event #2: Blocking Lodestone

General
Name: Blocking Lodestone
☐ Global Event

Page Options: New Page | Copy Page | Paste Page | Delete Page | Clear Page

Pages: 1 | 2

Conditions: Spawn/Execution Conditions

Entity Options
Animation: None
Movement
Type: None
Set Route...
Speed: Normal
Freq: Normal
Layer: Same as Play

Entity Inspector Options
Face: None
☐ Disable Inspector
Inspector Description:

Extras:
☐ Passable ☐ Hide Name ☐ Dir Fix ☐ Walking Anim
☐ Interaction Freeze

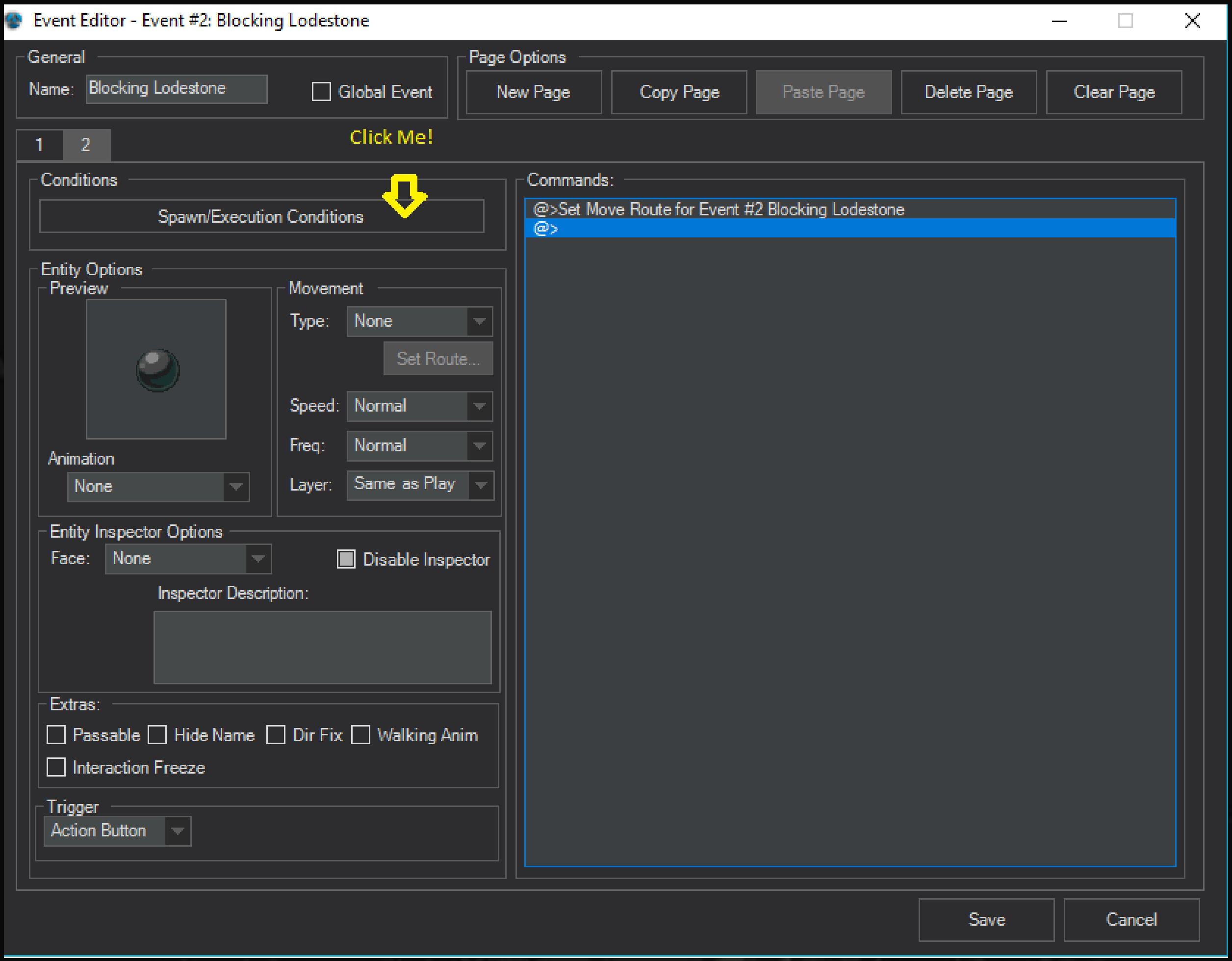
Trigger: Action Button

Commands:
```
@>Show Text: The rock is too heavy to push.
@>
```

Save | Cancel

Here is our Page 2. Now we need to create our lock (Condition) to keep our box nice and locked.

Because we only want a Warrior to be able to push the lock, no one else!



Event Editor - Event #2: Blocking Lodestone

General — Name: Blocking Lodestone ☐ Global Event

Page Options: New Page | Copy Page | Paste Page | Delete Page | Clear Page

Pages: 1 | 2 — Click Me!

Conditions: Spawn/Execution Conditions

Commands:

```
@>Set Move Route for Event #2 Blocking Lodestone
@>
```

Entity Options — Preview; Animation: None

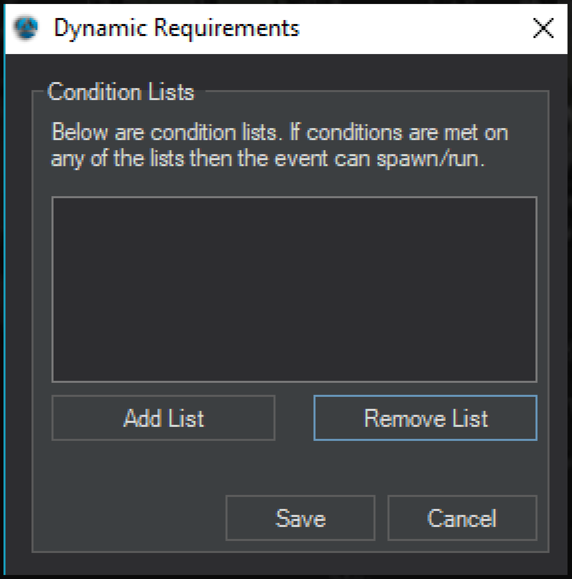
Movement — Type: None; Set Route...; Speed: Normal; Freq: Normal; Layer: Same as Play

Entity Inspector Options — Face: None; Disable Inspector; Inspector Description:

Extras: ☐ Passable ☐ Hide Name ☐ Dir Fix ☐ Walking Anim ☐ Interaction Freeze

Trigger: Action Button

Save | Cancel



Dynamic Requirements

Condition Lists

Below are condition lists. If conditions are met on any of the lists then the event can spawn/run.

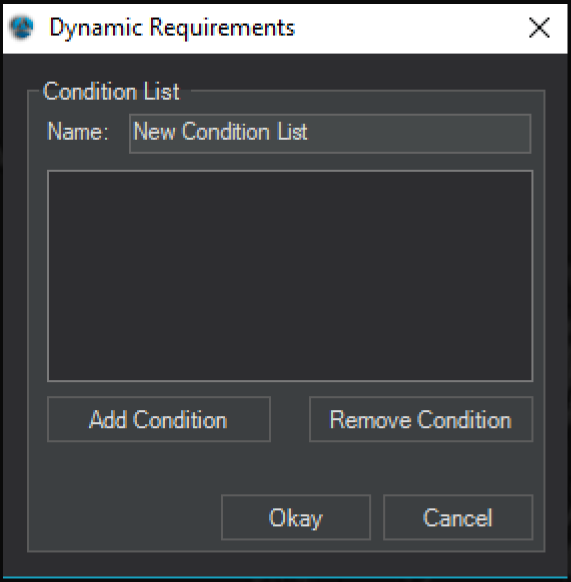
Add List | Remove List

Save | Cancel

This is the Condition box that we just opened. It is Important to note each Condition we place in here acts as a lock for our box. Only a specific key will fit the lock, and each lock we have must have a different key before it will open!

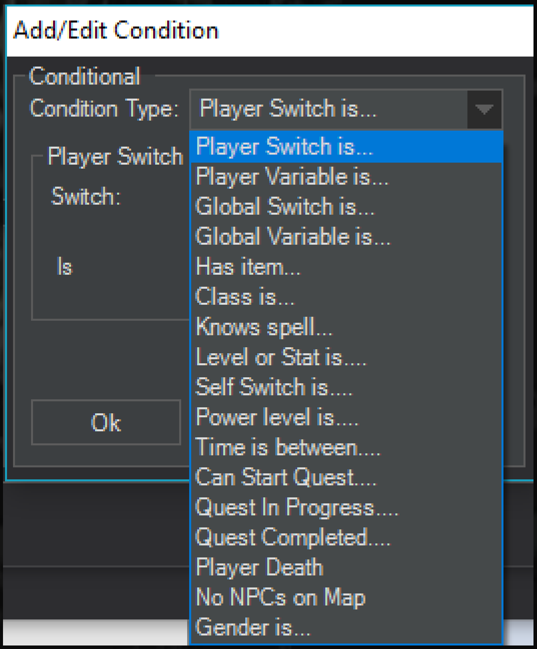
Now, it is time for us to lock our boxes. This will make Page 2 locked until the certain Conditions are met (the keys that fit the lock). To clarify, when we activate the Event, Page 2 will check if we are a Warrior. If we are, it will open run the Event Commands there, if not, it will keep to the previous page.

So, let's click *"Add List."*



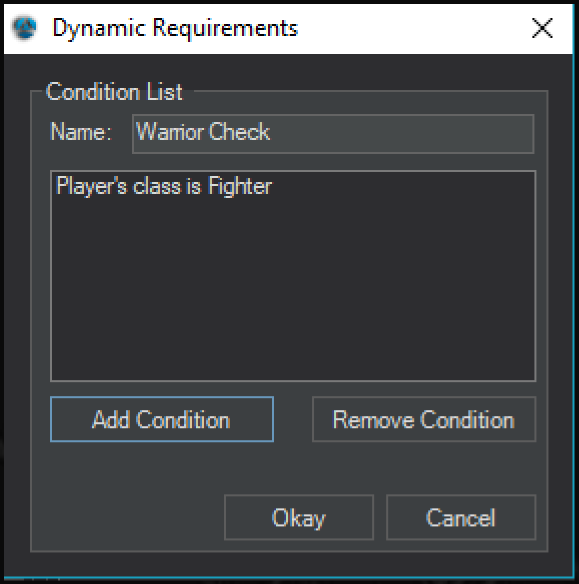
Now need to name our lock, and add as many Conditions as we need.

Condition (Lock) Masterlist.



As you can see, there are so many different locks we can Craft, and can place multiple locks on one chest. Remember Switches and Variables? BAM, this is Where we use it at. Does Marie love of us? Use the Player Variable is equal or greater than 3. If it is, The page loads and we get more dialogue! If not, she Ignores us. In our case though we are checking the class.

We named our Condition and chose our key: *“Player’s Class is Fighter.”*



Name: Warrior Check

Player's class is Fighter

Now the Event has two pages. The first page will run if we are not a Warrior; however, if we are of the Warrior class, Page 2 will run for us.

To open the box, we must have the key to the lock. The Engine simply asks:

Are you a warrior?

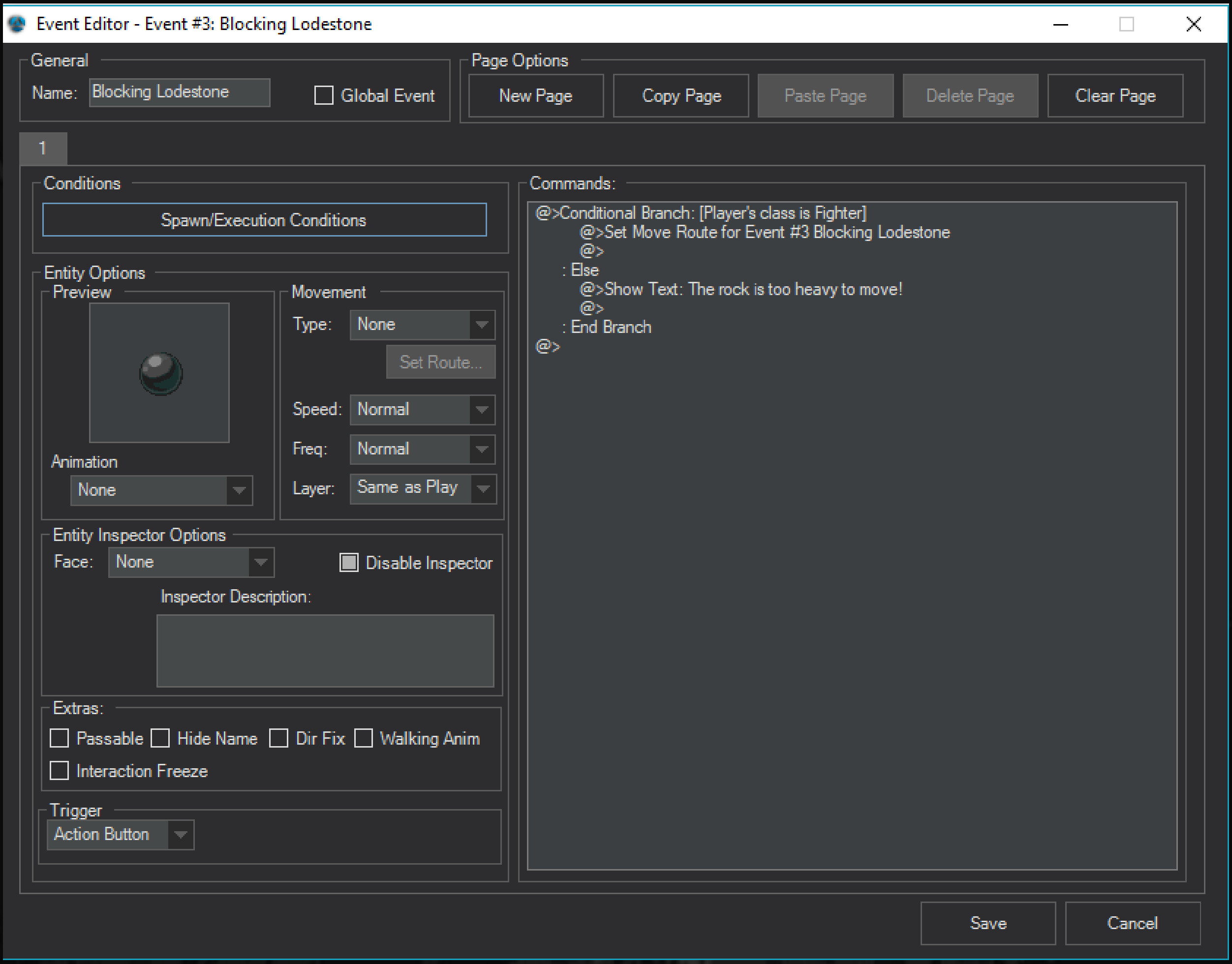
No = Page 1 activates and tells you the Rock is too heavy to push.

Yes = Page 2 activates and the Rock moves to the side for us.

The best part is we can keep adding more pages, and Conditions! We can keep going and have the rock explode if your class is a Spell Caster on page 3, or on page 4 you can destroy the rock if you have a pickaxe, or have it check the position of the Switch to see if you talked to a person yet. The possibilities are nearly endless.

Single Page Event Example

So, let's create our Event like our previous one; however, there are a few key differences. Notice we are only using 1 Page, instead of 2. I used the Event Command: Conditional Branch. What this does is take the Page Conditions we were looking at and puts it in the Event Command. Take a look:



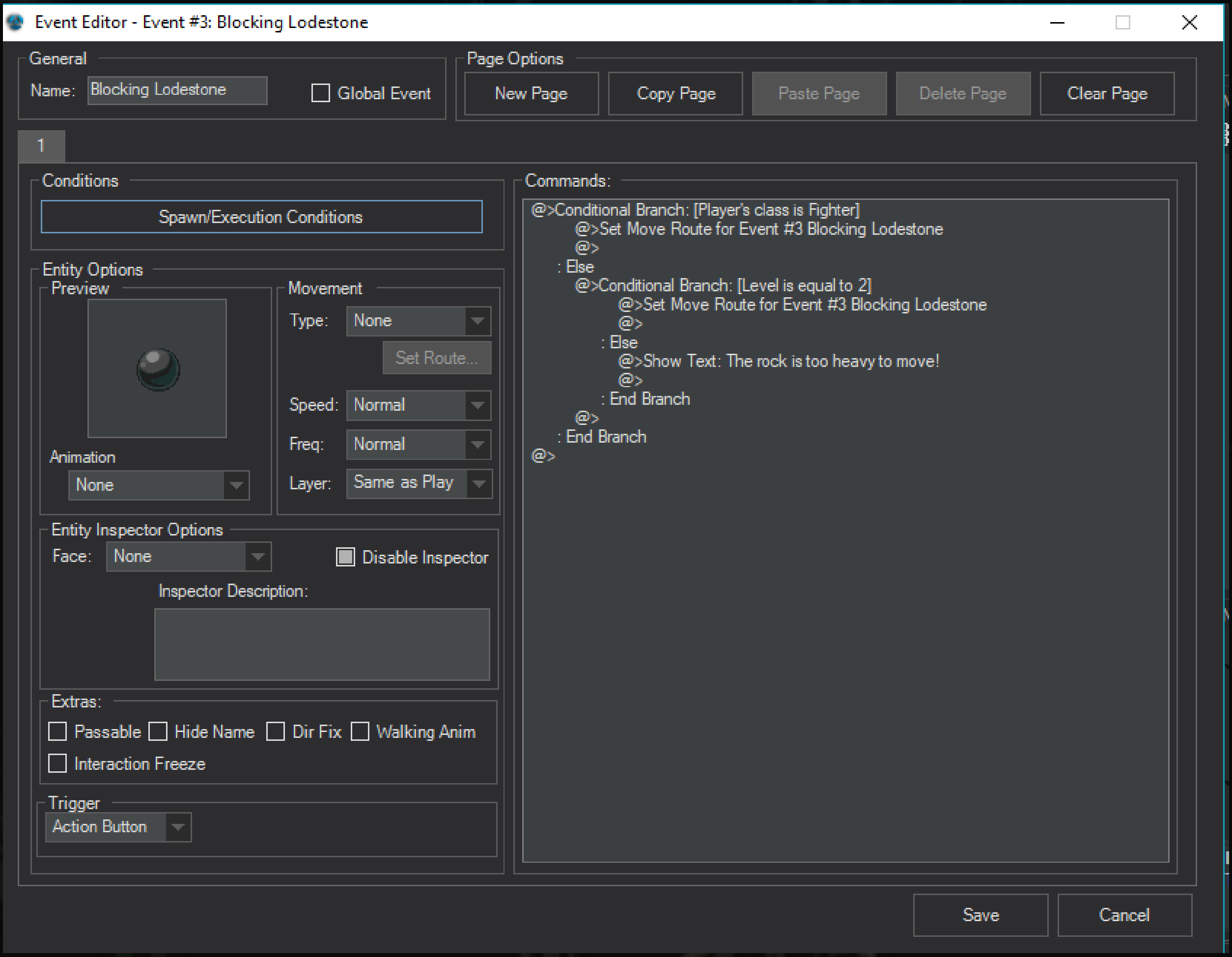
Type: None
Speed: Normal
Freq: Normal
Layer: Same as Play
Animation: None
Face: None
Disable Inspector
Trigger: Action Button

```
@>Conditional Branch: [Player's class is Fighter]
    @>Set Move Route for Event #3 Blocking Lodestone
    @>
  : Else
    @>Show Text: The rock is too heavy to move!
    @>
  : End Branch
@>
```

We set up the Conditional Branch in similar fashion: When we activate the Event, the Conditional Branch checks if we are a Fighter or not. Reference our earlier conversation on Conditional Branches if you need a reference how this is read. It works exactly the same way as our previous example, but was done in half the time, so that brings me to our next point. Choose the best tool for the job. Do you need to have multiple pages or can you simply use one page and create a complex Conditional Branch Tree? Why use a sledgehammer to pound a nail in, when a simple hammer would suffice?

Like the lists we can made in our previous example, we can actually put Conditional Branches Inside Conditional Branches. This is essentially putting a locked box, inside a locked boxed, within a locked box or placing more locks or keyholes on the box.

Multiple Locks and Keyholes



```
@>Conditional Branch: [Player's class is Fighter]
    @>Set Move Route for Event #3 Blocking Lodestone
    @>
  : Else
    @>Conditional Branch: [Level is equal to 2]
        @>Set Move Route for Event #3 Blocking Lodestone
        @>
      : Else
        @>Show Text: The rock is too heavy to move!
        @>
      : End Branch
    @>
  : End Branch
@>
```

Are you a Warrior, no? What about your level? Yes!? Then let's open!

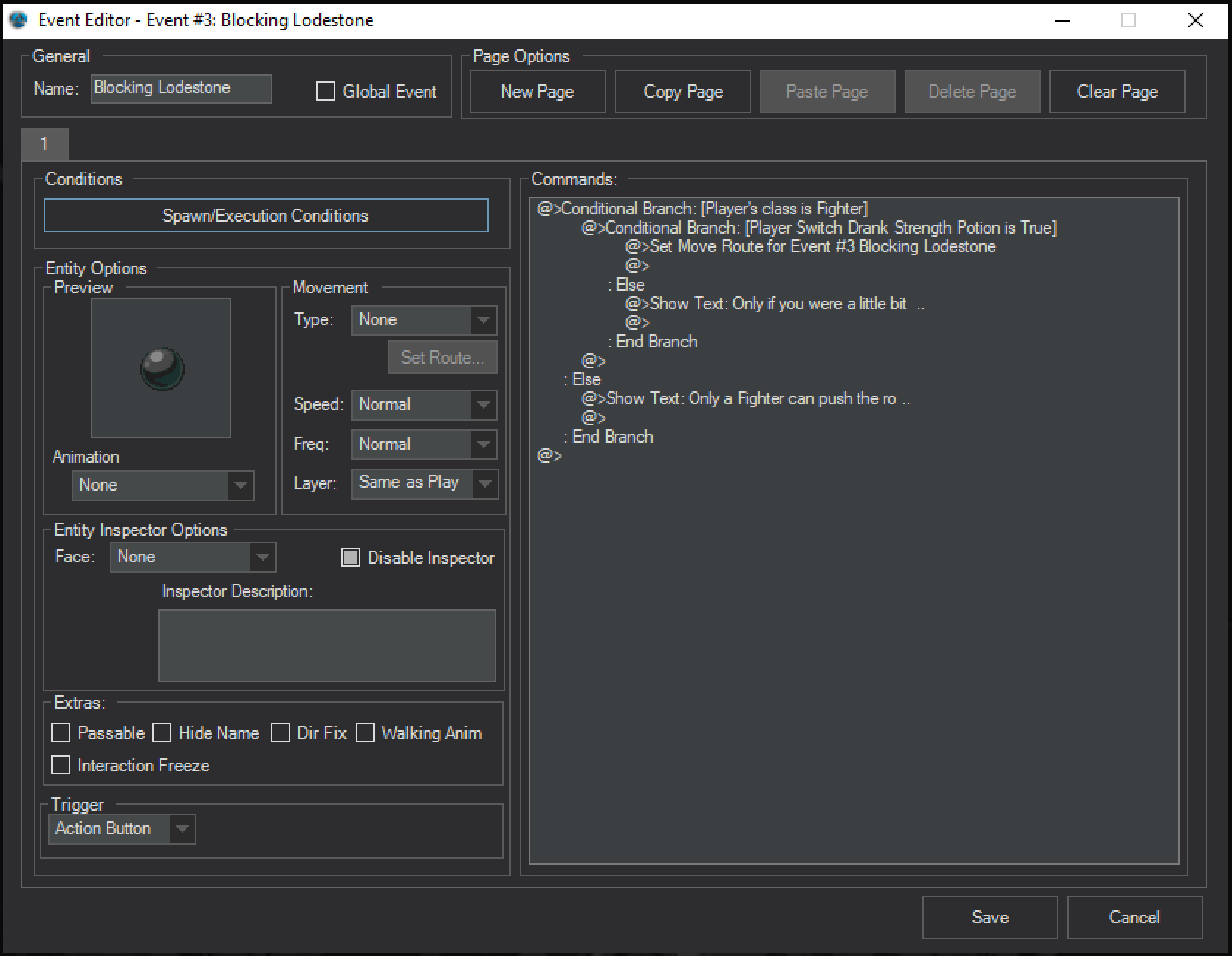
Locked Boxes Inside Locked Boxes

Are we a fighter? No? say it is too heavy.

Are we a fighter? Yes! then check if we drank the Strength Potion.

Did we drink? No? say we need to be stronger.

Did we drink? Yes! Move the rock!



```
@>Conditional Branch: [Player's class is Fighter]
    @>Conditional Branch: [Player Switch Drank Strength Potion is True]
        @>Set Move Route for Event #3 Blocking Lodestone
        @>
      : Else
        @>Show Text: Only if you were a little bit  ..
        @>
      : End Branch
    @>
  : Else
    @>Show Text: Only a Fighter can push the ro ..
    @>
  : End Branch
@>
```

As we have seen, there are multiple ways to do different things. Your job as a Game Developer is to utilize the best tool for the job and make it not only work, but understandable when you or someone else looks at the Event/Event Commands.

In Closing…

Whew, we got through it! I know there was quite a bit in this document; however, we have gone through almost everything (the basics, all the way up to more complex concepts of the editor). Be sure to keep practicing and do not be afraid to try something new to see how it works. Working in the Event Editor will become second nature to you! Keep this document close to you as a refence guide when you are working on your game within the Intersect Engine. I hope it helps you become a better game developer and look forward to future tutorials!

Thank you,

Agoraphobic (Jamie Hornsby)

Labyrinthhearts84@gmail.com

http://ascensiongamedev.com/