2. Registering the Event

Here I show you how to register your eventh{pt .}c!pp, so the game is able to call the event and integrate it into it's systems.

On the last page we created the Event **event_label**.

Now before registering, we need to decide, where the Event will be stored.

For that the game uses a class called **EventStorage**. This class registers the events and also provides them when called.
Currently there are 3 types of **EventStorage,** the EventStorage, the TempEventStorage and the FragmentStorage.
We'll ignore the FragmentStorage for now.

A more detailed explanation of the EventStorages follows in a later chapter.

Now back to the question where we want to register the Event. Tha game has multiple locations, for example the school building, the courtyard and so on. Now each of those locations has three Storages. a general one, a TempEventStorage and a dictionary of multipal EventStorages. The difference between those is, that Events registered in the TempEventStorage will only be shown once and then never again in the playthrough, while Events stored in the normal EventStorage can be shown repeatedly.

When a location is entered by the player, the game will first try to call an event from the TempEventStorage. After that it will try to open an Event from the general EventStorage, and after that it will open a Selection Menu, where the player can select the action he want's to execute. So each action is it's own EventStorage.

In this example, I will register the event for the courtyard, and as it should be shown repeatedly, we use the normal Storage.
So the registration will look like this:

```
init 1 python:
    set_current_mod('base')


    event_label_event = Event(3, "event_label")
    courtyard_events["patrol"].add_event(event_label_event)
```

So what do we see here. First we open a python block that runs when the game loads. The **1** shows the position in the load order. When registering events, the position should not be lower than 1.

In the second row we set the current mod. This is important for the game to detect from which mod the registrations come from. It is your responsibility to make sure, that when you register content into the game, that you call that method with your mod key at least once at the beginning of each init block. Make sure that the key is identical to the key you provide in the metadata file.

In row 4 we define the event we wrote in the last page of this manual. It is simply built by initializing the Event class with a priority and the renpy label it has to call.

And in row 5 we register the event. As you can see, we add the event into the EventStorage for the patrol action in the courtyard.

With that the event is sucessfully registered and will now be able to be called by the game. But since we added only the barebone of informations, it willl work, but there will be no image in the replay gallery and there is no limitation when it can be called. With that there is always a chance that it can be called.

A list of all Storages can be found here. And how to add your own storages can be found here.

In the next page we will work on how to set conditions and limitations on when an event is allowed to be called.

Event(priority: int, event: str, *options: Condition | Selector | Option | Pattern, thumbnail: str = "", register_self: bool = True, override_intro: bool = False, override_location: bool = None)

This class represents a callable game event.

This class performs a self-check when initialized and will not be registered when something is not correct. It will also throw an error message in the log stating the problem.

priority

This value represents the way an EventStorage selects the event. There are three priorities ranging from 1 to 3. Some methods that check for or call events can also use priority 0 which then represents all three priorities.

Priority 1 events are high priority, that means the EventStorage elects the first viable Priority 1 Event and calls it. After that event is fnished, the game will return to the map overview.

Priority 2 events are also high priority but lower than 1. An EventStorage will collect all viable Priority 2 events and call them one after another and then return back to the map overview.

Priority 3 events are also referred to regular events. Here the EventStorage will collect all viable Priority 3 events and then select one at random to call it. After that event, the game will return to the map overview.

event

This value is the renpy label the event opens when called.

options

This is an argument list that can contain various numbers of Conditions, Selectors, Options and Patterns. These are used to modifiy an event and to give it more detail and restrictions on when an event is allowed to be called.

After this parameter, all following parameters have to be called by their parameter name

---

register_self

This is an option to disable or enable the event from being registered in the central event collection. This value is True by default, since an event should always be registered in the central event collection since different instances use that collection to recieve the events data. But rarely it is necessary to delay that registration so this option exists.

override_intro

By default all events are blocked from being called during the introduction phase of the game, since there are two free roaming phases, that run on the normal event system. To block the event from being run during those phases, it adds a Condition to itself that blocks the use during the introduction. To prevent that blocking from happening, this parameter can be set True. That would prevent the event from adding the condition and allow the event to be called during the introduction.

override_location

Normally an event gets it's location from the EventStorage it is added to, but in some cases it is needed to override the location. The location value is only important for the replay gallery and does not have any other role that to sort the event into the correct category in the gallery.

Revision #7
Created 17 September 2024 11:38:33 by SuIT-Ji
Updated 1 October 2024 07:42:07 by SuIT-Ji