1. Writing the Event

Here I explain, how you write the body of an event. c!pp

Creating an Event follows multiple Step.

Step 1 ist to write the Event itself.

An Event has to follow the following structure:

```
label event_label (**kwargs):
    $ begin_event(**kwargs);

    // get values

    // get images

    // dialogue or whatever happens during the event

    // stat changes

    $ end_event('new_daytime', **kwargs)
```

So the **label** defines where the event begins.
An event always takes a **kwargs parameter. In it the game delivers all the data and information the game, the event and other functions need to work. So you need to make sure to not overwrite or change values in it haphazardly.

When using the normal value system, to insert data in the event for you to use, there will be no collision since those are stored in a dedicated dict inside the kwargs.

Next is the **begin_event** method.

begin_event(version: str = "1", **kwargs)

This method is called at the start of an event after choices and topics have been chosen in the event.
    It prevents rollback to before this method and thus prevents changing choices and topics.

It also starts the Gallery_Manager if the event is not in replay which is used to track and register
the used variables and decisions in the event.

Parameters

1. version: str (Default "1")
   - The version of the event. This is used to identify the event version.

Options

1. no_gallery = True
   - Gallery_Manager will not be initiated and event will not be registered into the gallery

This one has to be called at the beginning of each event. This method does a multitude of tasks to prepare and make the event work.

It starts the Gallery_Manager, which is responsible for properly registering the event and all the used values and decisions into the replay gallery.

By adding a version, you can tell the gallery if the stored data about the event is still valid or not. If the version varies from the one used for the data already stored, the gallery wipes the data about the event from the gallery storage. This is necessary if you change the event enough for the data to not be valid anymore. This is the case for example when removing or adding values or changing the order of the values occurence. To get a more detailed explanation, you can look here.

The begin_event method also blocks the player from rolling back further than the start of an event.

At the end of each event we have the **end_event** method.

end_event(return_type: str = "new_daytime", **kwargs)

This method is called at the end of an event.
It returns to the map overview or calls a new daytime event.
In case of a replay, it returns to the journal.

Parameters:

1. return_type: str (Default "new_daytime")
   - The type of the return.

- If return_type is "new_daytime", a new daytime event is called.
- If return_type is "new_day", a new day event is called.
- If return_type is "none", nothing is called.

This one is also important. This method closes the event. In case of a Composite Event, it calls the next Fragment, in case of the event being played as a replay from the gallery, it returns you to the gallery menu and otherwise it returns you to the game.

By default it returns you to the map but progresses the daytime cycle by one.

By using "new_day", you can skip the entire day and by using anything else or "none", the method simply returns and you can use another way to return to the game or do something else.

It is in your responsibility to have the event return to the game properly when not using the end_event method for returning.

In the next Page I show you how to register the event so it can be used by the game.

Revision #6
Created 5 September 2024 11:52:46 by SuIT-Ji
Updated 1 October 2024 07:41:59 by SuIT-Ji