TUSC training guide

TUSC trainings are YAML files with multiple documents in it. You can name the file whatever you want, just ensure to have `.yml` or `.yaml` extension.

Training schema

First step is to tell TUSC how to evaluate the training:

```yaml
schema: tusc.training:v2
requirements:
  stage_points: 0
```

Regarding values:

- `schema` : this tells TUSC how to load girlpack. Keep it untouched unless you know what you're doing
- `requirements` : an object containing various requirements for the training to be made.

Under `requirements` you can specify the following fields:

- `openness` : a number to require a minimum girl openness
- `stage_points` : a number of minimum training points over the specific girl; mind these points are referred to the training stage, not the overall girl openness
- `count` : a number to require a minimum number of trainings performed with the girl
- `max_count` : the maximum number of trainings performed with the girl for the training
- `group` : the minimum performance group number (0 - 4, where 0 is dancing, 4 is sex) to require
- `max_group` : the maximum performance group number (0 - 3, where 0 is dancing, 3 is oral) for the training
- `stage` : the minimum performance stage number (0 - 2, where the three values represent the sub-categories of every performance group, as in dance would be basic, closer, topless) to require
- `max_stage` : the maximum performance stage number (0 - 1, where the values represent the sub-categories of every performance group, as in dance would be basic, closer) for

the training

- `current_performance` : the current unlocked performance group (can be `dance` , `pose` , `foreplay` , `oral` , `sex`) to require
- `target_performance` : the next unlockable performance group (can be `dance` , `pose` , `foreplay` , `oral` , `sex`) to require

> **Note:** mind that girlpacks might not include some performances groups or stages, and thus the combination of requirements you put together might exclude the training for those girls.

Flow

Then you can start writing your personal *flow*. A flow is practically speaking a list of storylines or choices to make:

```
flow:
  - story:
      subject: girl
      text: >-
        Hey boss! How you doing?
  - story:
      subject: player
      text: >-
        Hey {girl_name}, all good, what about you?
```

As you can see a storyline requires a `subject` (girl or player), and the text to show.

Choices

Choices requires a bit of explaination:

```
flow:
  - story:
      subject: player
      text: >-
        (What should I do?)
  - choice:
      - text: Promise her more money
        compare: boasting
      - text: Say she will have a lot of fun
```

```
      compare: lewdness
    - text: Remind her who is in charge
      compare: obedience
```

As you see choices should contain a `text` like the story element, and a field of the girl to compare with (and thus decide if the action succeed or not). Valid comparison are:

- lewdness
- boasting
- obedience

You might also specify the same parameters in a *force* matter, like `obedience:force`, which will use a different probability check considering the player `charisma` attribute.

Targets

Whenever you define a choice, you should also define the `success` and `failure` targets, so the training can continue depending on the result of the choice taken by player. To define a target, write after your flow (mind the initial `---`):

```
---
target: success
outcome:
  progress: true
  scene: true
  energy: -1

flow:
  - story:
      subject: girl
      text: >-
        here the story continue..
```

The `target` field is the identifier of the specific target. The default values are `success` and `failure`. You can define several targets for your choices, eg:

```
  - choice:
      - text: Say she will have a lot of fun
        compare: lewdness
        target:
          success: success_lewd
```

```
      failure: failure_lewd
  - text: Promise her more money
    compare: boasting
    target:
      success: success_boast
      failure: failure_boast
  - text: Remind her who is in charge
    compare: obedience
    target:
      success: success_obey
      failure: failure_obey
```

Then you will need to define the relvant targets with the same names.

The `outcome` section might contain:

- `progress` : a boolean (`true` or `false`) that specifies if the player can progress to next training stage/group
- `energy` : a number that specify how girl energy change (to lower girl energy you should use a negative number, eg `-2`)
- `mood` : a number that specify how girl mood change (to lower girl mood you should use a negative number)
- `salary` : a number (0-whatever) representing the salary increase in percentage (100 means double up salary)
- `scene` : a boolean (`true` or `false`) which tells TUSC if has to show a video at the ending of the training or not.

Mind that you can keep proceeding with other flows even in targets, just defines new choices and relative targets.