

HAUSTIER-SIMULATOR

VON LEICHT NACH SCHWER

AKTIONEN

Erstelle Funktionen, wie:

- feed()
 - Gibt eine lustige Reaktion zurück (z. B. 'Das Haustier hat deinen Snack geklaut.')
- play()
 - Gibt eine zufällige Spielaktion zurück.
- sleep()
 - Gibt einen Einschlaf-Spruch zurück.

Baue ein Menü, das diese Funktionen aufruft.

EREIGNISSE

neue Funktionen:

- random_event(chance=0.3)
 - Mit 30% Wahrscheinlichkeit passiert etwas Chaotisches.
 - Beispiel: 'Dein Haustier hat ein WLAN-Passwort geändert.'
- apply_event(state, event)
 - Verändert den Zustand entsprechend.

MINI-GAME

Baue ein Spiel mit:

- Rundenlogik
- Zustandsveränderungen
- Zufallsereignissen
- Sieg- und Niederlagebedingungen
- Humorvollen Texten
- Beispiel:
 - Sieg: 'Dein Haustier ist glücklich, satt und hat nichts angezündet.'

Struktur:

- start_game()
- play_round(state, round_number)
- check_end(state, round_number)
- end_game(result)

1

2

3

4

5

6

GRUNDFUNKTIONEN

Programmiere ein kleines Spiel, in dem ein eigenwilliges digitales Haustier betreut wird, das ständig Unsinn macht.

Implementiere:

- greet_pet(name): Begrüßt das Haustier
- pet_status(): Gibt einen zufälligen Zustand zurück, z. B. 'hungrig', 'schläfrig', 'hyperaktiv', ...

Mini-Programm:

- Haustier begrüßen
- Status anzeigen
- Ende

ZUSTANDSVERWALTUNG

Implementiere:

- get_pet_state()
 - Gibt ein Dictionary mit Werten wie Hunger, Energie, Laune zurück.
- update_state(state, action)
 - Verändert den Zustand besierend auf der Aktion.
- show_state(state)
 - Gibt den Zustand humorvoll aus.

Beispiel:

- Aktion 'spielen' senkt Energie und erhöht Laune-

ROBUSTE EINGABEN

neue Funktionen:

- safe_input(prompt)
 - Frag Eingaben ab und fängt Fehler ab.
- validate_action(action)
 - Prüft, ob die Aktion existiert.
- run_turn(state)
 - Führt einen kompletten Spielzug aus, inklusive Fehlerbehandlung.