Koszmar i skok

Koszmar i skok

Strona referencyjna. Pliki: `core/player/player.gd`, `core/state/game_state.gd`, `core/realm/nightmare_rules.gd`, `core/realm/nightmare_director.gd`, `core/realm/realm_manager.gd`, `core/realm/realm_environment.gd`, `content/realm/nightmare_rules.tres`. Opowieść i klocki: odcinek „Skok i sen, który czasem jest koszmarem”.

Skok (Player)

Element	Opis
Akcja `jump`	Spacja (klawisz fizyczny) i przycisk X pada (button 2).
`jump_velocity`	Prędkość pionowa na starcie skoku, domyślnie 4.5 m/s, czyli ok. 1.03 m wysokości ($v^2 / 2g$).
Zasady	Skok tylko z ziemi (`is_on_floor()`), bez podwójnego skoku. Przy wstrzymanym drzewie (menu) gracz nie przetwarza fizyki, więc skok nie działa.

Stan fabuły w GameState

Metoda	Działanie
`set_act(act)`, `get_act()`	Bieżący akt, od 1.
`set_quest_scope(scope)`, `get_quest_scope()`	Zakres questu, w którym liczymy koszmar (np. id aktywnego questu). Pusty = jeden wspólny zakres.
`mark_nightmare_used(scope)`, `was_nightmare_used(scope)`	Czy w danym zakresie był już koszmar.

Zapis: sekcja `story` (`act`, `quest_scope`, `nightmare_used`), `SAVE_VERSION = 3`. Zapisy v1 i v2 dostają wartości domyślne (akt 1, pusty zakres, brak użyć).

Reguły i losowanie

- `NightmareRules` (zasób): `chance_per_act: Array[float]` (indeks 0 = akt 1; akt spoza listy używa ostatniej wartości; pusta lista = nigdy), `chance_for_act(act) -> float`.
- `content/realm/nightmare_rules.tres`: wartości zastępcze 0.05, 0.10, 0.20, do ustalenia przez autora.
- `NightmareDirector` (`RefCounted`, **nie autoload**): `roll(rules) -> bool`.
  - Brak reguł albo koszmar już użyty w `quest_scope` = `false`, bez losowania.
  - W pozostałych przypadkach `rng.randf() < chance_for_act(act)`.
  - Wygrana zaznacza `mark_nightmare_used(quest_scope)`.
  - Jedna instancja żyje w `RealmManager.nightmare_director`. Pole `rng` (RandomNumberGenerator) można podmienić albo ustawić mu ziarno (testy).
  - Pole `force_next` (menu deweloperskie): następne losowanie na pewno da koszmar, bez zaznaczania użycia; kasuje się po jednym razie.
  - Dlaczego nie autoload: nie ma własnego stanu (stan trzyma `GameState`), więc globalna nazwa niczego by nie dała.

Koszmar w RealmManager

Element	Opis
`enum Realm { WAKING, DREAM, NIGHTMARE }`	Identyfikator koszmaru w GameState i RealmAware: `nightmare`.
`enter_dream(dream_id, kind = Realm.DREAM)`	`kind = Realm.NIGHTMARE` wchodzi w koszmar: flaga `nightmare_<id>_visited`, sygnał `nightmare_entered(dream_id)` (zamiast `dream_entered`). `false`, gdy gracz już śpi, trwa przejście albo `kind` to jawa.
`return_to_waking()`	Działa ze snu i z koszmaru, gracz wraca na `ReturnMarker` ostatniego punktu.
`is_asleep()`	Sen albo koszmar.

Wygląd i obiekty

- `RealmEnvironment.nightmare_look`: wygląd koszmaru. Puste = `dream_look`.
- `RealmAware.realms`: `[&"nightmare"]` = tylko koszmar, `[&"dream", &"nightmare"]` = sen i koszmar. Sam `[&"dream"]` w koszmarze jest ukryty.

Menu medytacji

„Zaśnij” wywołuje `RealmManager.nightmare_director.roll(point.nightmare_rules)` i wchodzi w koszmar albo zwykły sen. Etykieta jest zawsze ta sama, gracz nie wie wcześniej, co wypadnie. „Obudź się” jest widoczne we śnie i w koszmarze. Ognisko bez `nightmare_rules` daje zawsze zwykły sen.

Pułapki

- Spacja jest też w `dialogic_default_action` (przewijanie dialogów). Dialogic nie wstrzymuje gry, więc przy rozmowach trzeba będzie blokować skok.
- Test, który oczekuje zwykłego snu przy ognisku z regułami koszmaru, musi wcześniej zaznaczyć koszmar jako użyty (albo ustawić ziarno), inaczej będzie losowo padał.

Testy

`tests/test_player_jump.gd`, `tests/test_nightmare.gd`, `tests/test_state.gd`, `tests/test_main_scene.gd`, `tests/test_realm_rooms.gd`.

Opublikowano z repozytorium Horrorek (commit 87717ad) skryptem publikuj-bookstack.ps1.

Revision #2
Created 2026-10-28 00:48:13 UTC by Tomasz Konieczny
Updated 2026-10-28 22:53:57 UTC by Tomasz Konieczny