Odcinek: Skok i sen, który czasem jest koszmarem

Odcinek: Skok i sen, który czasem jest koszmarem

> Szkic do BookStacka. Faza 1. Poziomy: **[A]** klikane w edytorze, **[B]** gotowy skrypt do wklejenia, **[C]** skrypt z komentarzami dydaktycznymi. Pliki w repozytorium: `core/player/player.gd`, `core/realm/nightmare_rules.gd`, `core/realm/nightmare_director.gd`, `core/realm/realm_manager.gd`, `content/realm/nightmare_rules.tres`, `content/realm/room_test_nightmare.tres`, testy `tests/test_player_jump.gd`, `tests/test_nightmare.gd`.
>
> **Zmiana (zadanie 005B):** pokój testowy `content/rooms/room_test.tscn` i test `tests/test_room.gd` zastąpiły pokoje A, B, C (`content/rooms/room_a.tscn`, `room_b.tscn`, `room_c.tscn`, scena główna `content/main.tscn`) i testy `test_main_scene.gd`, `test_realm_rooms.gd`, `test_quest_rooms.gd`. Odcinek opisuje stan z chwili powstania.

Opowieść

Kapsuła nauczyła się skakać. Spacja albo X na padzie i podrywa się na niecały metr, ląduje, może skoczyć znowu. W powietrzu drugie naciśnięcie nic nie daje, bo tu nikt nie lata.

Potem wraca do ogniska i wybiera **„Zaśnij”**. Ten sam napis co zawsze. Ekran gaśnie, rozjaśnia się… i coś jest nie tak. Mgła jest gęstsza i czerwona. Ściany mają kolor snu, ale światło prawie zgasło. Monolitu, który zwykle stoi we śnie na środku pokoju, nie ma. Zamiast niego pod sufitem wisi coś, co wygląda jak oko.

To był **koszmar**, odmiana snu. Gracz nie wiedział, że go dostanie: menu niczego nie zapowiadało. Ognisko tylko rzuciło kością. W pierwszym akcie szansa jest mała, w późniejszych rośnie. W tym samym queście koszmar już się nie powtórzy. Następne zaśnięcie da zwykły sen, aż fabuła ruszy dalej.

„Obudź się” działa tak samo jak ze zwykłego snu: gracz staje przy ognisku, przy którym zasnął.

Pojęcie: losowanie z regułą

> **Losowość w grze to zawsze liczba losowa plus reguła, która mówi, kiedy wolno losować.**
>
> - **Szansa** to liczba od 0 do 1. Losujemy liczbę z tego samego zakresu: jeśli wypadła mniejsza niż szansa, zdarzenie zachodzi. Szansa 0 to „nigdy”, szansa 1 to „zawsze”.
> - **Reguła** mówi, czy w ogóle losujemy. U nas: koszmar najwyżej raz na quest. Jeśli w bieżącym zakresie questu już był, nie losujemy, tylko od razu wybieramy zwykły sen.
> - **Pamięć** to miejsce, w którym reguła zapisuje „już było”. U nas `GameState`, więc informacja trafia do zapisu gry i nie znika po wczytaniu.
> - **Ziarno** (seed) sprawia, że generator liczb losowych daje zawsze ten sam ciąg. Gra używa losowego ziarna, a testy stałego, żeby wynik był powtarzalny.
>
> Zasada kciuka: **szansę ustawia autor, regułę i pamięć pilnuje kod, ziarno ustawiają testy.**

Klocek 1 [A]: szansa koszmaru i wygląd koszmaru

1. Otwórz `content/realm/nightmare_rules.tres`. W polu **Chance Per Act** jest lista szans dla kolejnych aktów (pierwsza pozycja to akt 1). Obecne wartości 0.05, 0.10, 0.20 to zastępstwo do ustalenia przez autora. Akt spoza listy używa ostatniej wartości.
2. Zaznacz ognisko (`MeditationPoint`) i w polu **Nightmare Rules** wskaż ten plik. Puste pole oznacza, że przy tym ognisku nigdy nie wypadnie koszmar.

3. Utwórz zasób **RealmLook** dla koszmaru lokacji, np. `content/realm/akt1_las_nightmare.tres` (zwykle mniej światła, czerwieńsza i gęstsza mgła), i wskaż go w polu **Nightmare Look** węzła `RealmEnvironment`. Puste pole oznacza wygląd zwykłego snu.
4. Obiekt tylko z koszmaru: węzeł z `realm_aware.gd` i **Realms** = `nightmare`. Obiekt ze snu, który ma być też w koszmarze: **Realms** = `dream`, `nightmare`. Sam `dream` w koszmarze się nie pokaże.

Klocek 2 [B]: losowanie koszmaru

Gotowy plik: `res://core/realm/nightmare_director.gd`. Cała logika mieści się w jednej funkcji:

```
func roll(rules: NightmareRules) -> bool:
	if rules == null:
		return false
	var scope: StringName = GameState.get_quest_scope()
	if GameState.was_nightmare_used(scope):
		return false
	var chance: float = rules.chance_for_act(GameState.get_act())
	if chance <= 0.0:
		return false
	var hit: bool = rng.randf() < chance
	if hit:
		GameState.mark_nightmare_used(scope)
	return hit
```

Menu medytacji woła ją przy „Zaśnij”, a potem `RealmManager.enter_dream(dream_id, Realm.NIGHTMARE albo Realm.DREAM)`.

Klocek 3 [C]: skok tylko z ziemi

Wersja z komentarzami dydaktycznymi. Pełny kod jest w `res://core/player/player.gd`.

```
func _physics_process(delta: float) -> void:
	# 1. W powietrzu działa grawitacja: prędkość pionowa co klatkę maleje o g · delta.
	if not is_on_floor():
		velocity.y -= _gravity * delta
	# 2. Skok tylko na ziemi. Ponieważ to "elif", w powietrzu ta gałąź się
	#    nie wykona, więc podwójny skok jest niemożliwy bez dodatkowego licznika.
```

```
    #    is_action_just_pressed = tylko w klatce naciśnięcia, nie przy trzymaniu.
    elif Input.is_action_just_pressed(&"jump"):
        # 3. Nadajemy prędkość w górę. Wysokość wyniesie ok. v² / (2g):
        #    4.5² / (2 · 9.8) ≈ 1.03 m.
        velocity.y = jump_velocity
    # ... dalej ruch w poziomie i move_and_slide() ...
```

Do przemyślenia: gdy menu wstrzymuje grę, `_physics_process` gracza w ogóle się nie wykonuje, więc Spacja w menu nie wywoła skoku. Co się stanie z Dialogic, który też używa Spacji, a nie wstrzymuje gry?

Sprawdź, że działa

Z katalogu głównego projektu, w PowerShellu:

```
.\tools\run-tests.ps1
```

Oczekiwane: w tabeli każdy wiersz ma `Wynik = OK`, także `test_player_jump` i `test_nightmare`.

Ciekawostka z kulis

Zmierzony skok w teście ma 1.07 m, a wzór daje 1.03 m. To nie błąd: Godot liczy ruch krokami co 1/60 sekundy, a nie w sposób ciągły, więc szczyt wypada trochę wyżej. Dlatego test porównuje wynik ze wzorem z tolerancją ±8 cm.

Druga ciekawostka: w teście pokoju „Zaśnij” musi dawać zwykły sen, ale ogniska mają już reguły koszmaru z szansą 5%. Raz na dwadzieścia uruchomień test by padał. Zamiast wyłączać losowanie, test zaznacza koszmar jako już użyty w bieżącym queście, czyli korzysta z tej samej reguły, którą testuje `test_nightmare`.

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

Revision #2
Created 2026-10-28 00:48:05 UTC by Tomasz Konieczny
Updated 2026-10-28 22:53:46 UTC by Tomasz Konieczny