Odcinek: Ognisko, sen i druga wersja tego samego pokoju

Odcinek: Ognisko, sen i druga wersja tego samego pokoju

> Szkic do BookStacka. Faza 1. Poziomy: **[A]** klikane w edytorze, **[B]** gotowy skrypt do wklejenia, **[C]** skrypt z komentarzami dydaktycznymi. Pliki w repozytorium: `core/state/game_state.gd`, `core/realm/`, `core/meditation/`, `core/ui/fade_overlay.tscn`, `content/realm/`, `content/rooms/room_test.tscn`, testy `tests/test_state.gd`, `tests/test_realm.gd`, `tests/test_meditation.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

W pokoju testowym płoną dwa małe ogniska. Kapsuła podchodzi do pierwszego i na ekranie pojawia się **„Medytuj”**.

Gra zamiera, a na środku ekranu wyskakuje krótka lista: **„Odpocznij < 1 godz. >”**, **„Zaśnij”** i **„Wróć”**. Strzałką w prawo gracz ustawia siedem godzin i wybiera odpoczynek. Zegar w pamięci gry przeskakuje z 20:00 na 3:00, a dzień zmienia się z pierwszego na drugi.

Przy ognisku gracz wybiera **„Zaśnij”**. Ekran gaśnie, a po chwili się rozjaśnia. To ten sam pokój, ale ściany mają kolor zaschniętego wina, w powietrzu wisi mgła, stół zniknął, a na środku stoi czarny monolit z czerwoną poświatą. Gracz nie przeniósł się do innej sceny: zmieniła się **rzeczywistość**.

Drugie ognisko stoi przy południowej ścianie. Tam „Zaśnij” jest wyszarzone i brzmi **„Nie możesz teraz zasnąć”**. Odblokuje się dopiero wtedy, gdy gracz otworzy drzwi do piwnicy kluczem z poprzedniego odcinka.

We śnie menu ogniska ma tylko jedną drogę: **„Obudź się”**. Ekran znów gaśnie i gracz stoi przy ognisku, przy którym zasnął, twarzą do płomienia.

Pojęcie: stan świata i dwie rzeczywistości

> **Stan świata to wszystko, co gra musi pamiętać, żeby po wczytaniu zapisu wyglądać tak samo.**
>
> - U nas mieszka on w jednym miejscu: w autoloadzie `GameState`. Poza przedmiotami i flagami są tam teraz **godzina**, **dzień**, **rzeczywistość** (`waking` albo `dream`) i **ostatni punkt medytacji**.
> - Wszystko inne tylko ten stan **odzwierciedla**. `RealmManager` nie decyduje, czy jest sen. Zapisuje to w `GameState` i ogłasza zmianę sygnałem `realm_changed`.
> - Dzięki temu zapis gry to jeden słownik z `GameState.serialize()`. Po wczytaniu (`deserialize()`) świat sam przełącza się na właściwą rzeczywistość, bez animacji.
>
> **Dwie rzeczywistości to dwie wersje tej samej lokacji, nie dwie sceny.**
>
> - Węzły **bez** komponentu `RealmAware` są wspólne: podłoga, drzwi, ogniska.
> - Węzły **z** `RealmAware` istnieją tylko w wybranych rzeczywistościach. Poza nimi są niewidoczne, nie działają i nie kolidują.
> - Wygląd lokacji (tło, światło, mgła) opisują dwa zasoby `RealmLook`, które `RealmEnvironment` podmienia przy każdej zmianie rzeczywistości.

> Zasada kciuka: **GameState pamięta, RealmManager przełącza, RealmAware i RealmEnvironment się dostosowują.**

Klocek 1 [A]: ognisko w nowej lokacji

1. Przeciągnij do sceny `res://core/meditation/meditation_point.tscn`.
2. W Inspektorze ustaw:
   - **Point Id**: stały identyfikator, np. `meditation_akt1_las` (bez polskich znaków),
   - **Dream Id**: identyfikator snu tej lokacji, np. `akt1_las`,
   - **Dream Available**: odznacz, jeśli przy tym ognisku nie da się zasnąć,
   - **Required Flag**: flaga fabularna, np. `door_cellar_open` (puste = brak warunku).
3. Rozwiń węzeł i przesuń **ReturnMarker** tam, gdzie gracz ma stanąć po przebudzeniu. Niebieska oś markera (−Z) to kierunek, w którym gracz będzie patrzył.
4. Uruchom scenę (F6), podejdź do ogniska i naciśnij przycisk interakcji.

Klocek 2 [A]: wersja snu lokacji

1. Utwórz dwa zasoby **RealmLook** w `content/realm/` (prawy klik → *Nowy zasób* → `RealmLook`), np. `akt1_las_waking.tres` i `akt1_las_dream.tres`. W wersji snu zwykle zmniejsza się **Light Energy** i **Ambient Energy** oraz włącza **Fog Enabled**.
2. Dodaj do sceny węzeł **Node**, przypnij mu skrypt `res://core/realm/realm_environment.gd`, wskaż oba zasoby oraz ścieżki do **WorldEnvironment** i **DirectionalLight3D** lokacji.
3. Obiekt tylko ze snu: dodaj **Node3D**, przypnij skrypt `res://core/realm/realm_aware.gd`, w polu **Realms** zostaw `dream`. Wszystko, co włożysz pod ten węzeł (meshe, kolizje, CSG, obszary), pojawi się tylko we śnie.
4. Obiekt tylko z jawy: to samo, ale w **Realms** wpisz `waking`.
5. Inny wygląd ścian we śnie: zrób dwa zestawy ścian, każdy pod własnym `RealmAware` (`waking` i `dream`), tak jak `WallsWaking` i `WallsDream` w `room_test.tscn`.

Pułapka: nie wkładaj `RealmAware` pod inny `RealmAware`. Zagnieżdżony węzeł jest pomijany przy wyłączaniu kolizji rodzica.

Klocek 3 [B]: własna pozycja w menu ogniska

Menu buduje się z listy pozycji, więc nowy system dopisuje się do niego bez zmian w `meditation_menu.gd`. Przykład: skrypt sceny, który do każdego ogniska dodaje pozycję „Wróć” w innym miejscu listy (w kolejnych zadaniach zastąpią ją leczenie, zapis i awans):

```
extends Node
## Dopisuje pozycję do menu wszystkich ognisk w scenie.

func _ready() -> void:
	for point: Node in get_tree().get_nodes_in_group(&"meditation_points"):
		point.menu_opened.connect(_on_menu_opened)



func _on_menu_opened(menu: MeditationMenu) -> void:
	menu.add_option(
		&"example",                      # id pozycji
		&"MEDITATE_BACK",                # klucz tłumaczenia etykiety
		func() -> bool: return true,     # czy pozycja jest teraz dostępna
		menu.close,                      # co zrobić po wyborze
		{ "realms": [&"waking"] as Array[StringName], "order": 50 }
	)
```

Dodatkowe pola w ostatnim słowniku: `realms` (gdzie pozycja jest widoczna), `locked_label_key` (napis, gdy pozycja jest niedostępna), `on_adjust` (strzałki lewo/prawo), `label_suffix` (tekst doklejany do etykiety) i `order` (kolejność, „Wróć” ma 1000).

Klocek 4 [C]: jak RealmAware „wyłącza” obiekt

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

```
func _disable() -> void:
	_active = false
```

```
	# 1. Ukrywamy węzeł. Dzieci dziedziczą widoczność, więc znikają razem z nim.
	visible = false
	# 2. Wyłączamy przetwarzanie: _process i _physics_process przestają się wywoływać.
	#    Dzieci z process_mode = Inherit też stają.
	process_mode = Node.PROCESS_MODE_DISABLED
	# 3. Pułapka: wyłączone przetwarzanie NIE wyłącza kolizji. Niewidzialna
	#    ściana dalej by blokowała gracza. Dlatego przechodzimy po wszystkich
	#    potomkach z kolizjami i zerujemy je ręcznie.
	for node: Node in _collect_physics_nodes(self):
		if node is CollisionObject3D:
			var body: CollisionObject3D = node
			# Zapamiętujemy oryginały, żeby po powrocie przywrócić je co do bitu.
			var values: Dictionary = {
				"collision_layer": body.collision_layer,
				"collision_mask": body.collision_mask,
			}
			# Warstwa 0 = nikt mnie nie widzi; maska 0 = ja nikogo nie widzę.
			body.collision_layer = 0
			body.collision_mask = 0
			if node is Area3D:
				var area: Area3D = node
				values["monitoring"] = area.monitoring
				values["monitorable"] = area.monitorable
				# set_deferred: Godot nie pozwala zmienić monitoringu w trakcie
				# obsługi sygnału fizyki, więc zmiana czeka do końca klatki.
				area.set_deferred(&"monitoring", false)
				area.set_deferred(&"monitorable", false)
			_saved[node] = values
		elif node is CSGShape3D:
			# CSG ma własny przełącznik kolizji.
			var csg: CSGShape3D = node
			_saved[node] = { "use_collision": csg.use_collision }
			csg.use_collision = false
```

Do przemyślenia: przeciwnik ze snu to po prostu scena przeciwnika pod węzłem `RealmAware` z `realms = [dream]`. Co powinno się stać z przeciwnikiem, który goni gracza w chwili przebudzenia?

Haki na kolejne zadania

- **Pozycje menu (zadania 004–006).** Leczenie, zapis gry i awans dopisują się przez `MeditationPoint.menu_opened(menu)` i `menu.add_option(...)`, jak w klocku 3. Menu ich nie zna i nie trzeba go przerabiać. Pozycja „Wróć” ma `order = 1000`, więc nowe pozycje lądują nad nią.
- **Zapis przy ognisku.** `GameState.serialize()` obejmuje już godzinę, dzień, rzeczywistość i ostatni punkt (`SAVE_VERSION = 2`). Zapisy w wersji 1 wczytują się z wartościami domyślnymi.
- **Przeciwnicy we śnie.** Przeciwnik pod `RealmAware` z `realms = [dream]` na jawie jest niewidoczny, nie przetwarza AI i nie koliduje. Po powrocie do snu wszystko wraca do oryginalnych wartości. Reakcje na samo przejście (np. reset pościgu) można podpiąć do `RealmManager.realm_changing`.
- **Pora dnia.** `GameState.hour_changed` i `day_changed` czekają na systemy zależne od czasu, np. światło lokacji albo zdarzenia o północy.
- **Odwiedzone sny.** `RealmManager.enter_dream` ustawia flagę `dream_<id>_visited`, której questy mogą używać jako warunku.

Sprawdź, że działa

Z katalogu głównego projektu, w PowerShellu:

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

Oczekiwane: tabela, w której każdy wiersz ma `Wynik = OK` (także `test_realm` i `test_meditation`), a na końcu napis `Wszystkie testy zaliczone`.

Ciekawostka z kulis

Menu medytacji **pauzuje grę** (`get_tree().paused = true`), a mimo to działa, bo ma `process_mode = Always`. To samo dotyczy zaciemnienia i `RealmManager`. Dlatego „Zaśnij” najpierw zamyka menu (wznowienie gry), a dopiero potem zaczyna przejście w sen. Gdyby kolejność była odwrotna, sen zaczynałby się w zamrożonym świecie, a gracz po przebudzeniu stałby w pauzie.

Drugi szczegół: w zapisie godzina jest liczbą z ułamkiem (`float`). Flagi w GameState zamieniają „okrągłe” liczby z JSON na całkowite, ale godzina celowo tę konwersję omija. Bez tego 4:00 wróciłoby z zapisu jako liczba całkowita 4, a 4:30 jako ułamek, czyli raz jednym typem, raz drugim.

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

Revision #4
Created 2026-10-27 23:24:42 UTC by Tomasz Konieczny
Updated 2026-10-28 22:53:44 UTC by Tomasz Konieczny