# Odcinek: Sen, który za każdym razem jest inny

# Odcinek: Sen, który za każdym razem jest inny

> Szkic do BookStacka. Faza 1. Poziomy: **[A]** klikane w edytorze, **[B]** gotowy skrypt do wklejenia, **[C]** skrypt z komentarzami dydaktycznymi. Pliki w repozytorium: `core/dream/`, `content/dream/`, `content/rooms/room_d.tscn`, `core/realm/realm_manager.gd`, test `tests/test_dream_generator.gd`.

## Opowieść

Z pokoju A prowadzi nowe przejście, na drogę. Na jawie to zwykła, pusta polana: wydeptana droga ze wschodu na zachód, płot, namiot, ognisko i niebieskawy kamień, który ktoś tu postawił nie bez powodu.

Kapsuła zasypia przy ognisku. Kiedy ekran się rozjaśnia, polana jest pełna drzew. Między nimi leży sadzawka, a od drogi odchodzi wąska ścieżka, która skręca raz w lewo, raz w prawo i kończy się małym obozem: namiot, żarzące się palenisko. Ktoś tu był. Albo będzie.

Kapsuła się budzi i zasypia jeszcze raz. Teraz drzewa stoją gdzie indziej, ścieżka skręca w inną stronę, a sadzawek jest dwie. Za to droga, namiot, ognisko i niebieski kamień są dokładnie tam, gdzie zawsze. Sen zmienia to, co do niego należy, i nie rusza tego, co należy do świata.

A jeśli kapsuła we śnie wyjdzie na chwilę do pokoju A i wróci? Drzewa stoją tak samo jak przed wyjściem. To wciąż ten sam sen.

# Pojęcie: powtarzalna losowość (ziarno)

> **Komputer nie losuje naprawdę. Liczy kolejne „losowe” liczby ze wzoru, a punktem startowym jest ziarno.**
>
> - To samo ziarno daje zawsze ten sam ciąg liczb, a więc ten sam sen. Inne ziarno daje inny ciąg i inny sen.
> - Przy każdym zaśnięciu gra losuje **nowe ziarno**, więc sen jest niespodzianką. Ziarno trafia do zapisu gry, więc po wczytaniu sen jest ten sam.
> - Każdy slot ma **własny generator**, zasiany z trzech rzeczy: ziarna snu, nazwy pokoju i nazwy slotu. Dzięki temu dodanie nowego slotu nie przesuwa wszystkich drzew w pozostałych.
> - Generator **nigdy** nie używa „globalnej” losowości (`randf()`). Jedna taka linijka i sen po wczytaniu wyglądałby inaczej.
>
> Zasada kciuka: **losowo dla gracza, powtarzalnie dla komputera.**

# Klocek 1 [A]: nowy slot snu

1. Otwórz pokój i dodaj **Marker3D** ze skryptem `res://core/dream/dream_slot.gd` (najlepiej pod węzłem `Slots`).
2. Ustaw **Slot Id** (np. `vegetation_west`, bez polskich znaków) i **Slot Type** (Vegetation, Water, Camp albo Branch).
3. Strefa: dodaj dziecko **Area3D** z **CollisionShape3D** (pudełko albo kula) i rozciągnij ją na obszar slotu. Bez strefy liczy się **Radius**.
4. Wskaż **Dream Palette** (np. `content/dream/palette_dream_vegetation.tres`) i opcjonalnie **Nightmare Palette**.
5. Ustaw **Min Items**, **Max Items** i **Min Distance**.
6. Rzeczy, których sen nie może zasłonić: pod namiotem albo ogniskiem dodaj **Marker3D** ze skryptem `res://core/dream/dream_anchor.gd` i ustaw **Radius**; obiekt fabularny dodaj do grupy `story_critical`.
7. Pokój musi mieć węzeł **Node** ze skryptem `res://core/dream/dream_generator.gd`, a w polu **Road Path** wskazaną drogę (`Path3D`).
8. Uruchom grę, zaśnij i naciśnij F9. W sekcji „Sen proceduralny” włącz „Pokaż strefy slotów” i przeładuj sen nowym ziarnem.

# Klocek 2 [B]: ziarno slotu

Gotowy kod w `res://core/dream/dream_generator.gd`:

```
static func slot_seed(dream_seed: int, room: StringName, slot_id: StringName) -> int:
	return ("%d|%s|%s" % [dream_seed, room, slot_id]).hash()
```

i jego użycie przy każdym slocie:

```
var rng: RandomNumberGenerator = RandomNumberGenerator.new()
rng.seed = slot_seed(dream_seed, room_key, slot.slot_id)
```

# Klocek 3 [C]: czy tu wolno postawić drzewo?

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

```
func _is_free(point: Vector3, element_radius: float, same_slot: Variant, min_distance: float,
		ignore_group: String = "") -> bool:
	# 1. Droga główna: liczymy odległość od jej osi (Path3D). Pas road_clearance
	#    plus promień elementu musi zostać wolny, bo droga należy do warstwy stałej.
	if _road_distance(point) < road_clearance + element_radius:
		return false
	# 2. Punkty stałe (namiot, ognisko): każdy ma własny promień ochrony.
	for anchor: Node in get_tree().get_nodes_in_group(DreamAnchor.GROUP):
		if anchor is DreamAnchor and get_parent().is_ancestor_of(anchor):
			if _flat_distance(point, (anchor as DreamAnchor).global_position) < (anchor as
DreamAnchor).radius + element_radius:
				return false
	# 3. Obiekty fabularne: sen nie może zasłonić niczego, co jest potrzebne w queście.
	for story: Node in get_tree().get_nodes_in_group(STORY_CRITICAL_GROUP):
		if story is Node3D and get_parent().is_ancestor_of(story):
			if _flat_distance(point, (story as Node3D).global_position) < story_clearance +
element_radius:
				return false
	# 4. Inne elementy snu: nie nachodzą na siebie (suma promieni), a w tym samym
```

```
	#    slocie trzymają jeszcze min_distance. Kawałki tej samej odnogi mogą się stykać.
	for other: Dictionary in _occupied:
		if ignore_group != "" and other["group"] == ignore_group:
			continue
		var needed: float = float(other["radius"]) + element_radius
		if other["slot"] == same_slot:
			needed = maxf(needed, min_distance)
		if _flat_distance(point, other["position"]) < needed:
			return false
	return true
```

Do przemyślenia: ta funkcja sprawdza tylko rzeczy, które się nie ruszają. Co by się stało z powtarzalnością, gdyby brała pod uwagę pozycję gracza?

# Sprawdź, że działa

Z katalogu głównego projektu, w PowerShellu:

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

Oczekiwane: w tabeli każdy wiersz ma `Wynik = OK`, w tym `test_dream_generator`. W grze: pokój A, portal na drogę, zaśnięcie przy ognisku, F9 i sekcja „Sen proceduralny”.

# Ciekawostka z kulis

W liczniku menu deweloperskiego typy pokazywały się w przypadkowej kolejności: „water, vegetation, branch, camp”. Powód: sortowanie identyfikatorów `StringName` w Godocie nie porównuje tekstu, tylko wewnętrzne adresy. Pomogło sortowanie po zamianie na zwykły tekst. Ta sama pułapka groziłaby generatorowi, dlatego sloty są sortowane po `String(slot_id)`, a nie po samym `StringName`.

---

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

---

Revision #1
Created 2026-10-28 22:53:49 UTC by Tomasz Konieczny
Updated 2026-10-28 22:53:49 UTC by Tomasz Konieczny