Odcinek: To, co zostaje

Odcinek: To, co zostaje

> Szkic do BookStacka. Faza 1. Poziomy: **[A]** klikane w edytorze, **[B]** gotowy skrypt do wklejenia, **[C]** skrypt z komentarzami dydaktycznymi. Pliki w repozytorium: `core/combat/death_director.gd`, `core/combat/remnant.tscn`, `core/combat/remnant_keeper.gd`, `core/stats/sanity_rules.gd`, `core/ui/death_screen.tscn`, `content/combat/death_rules.tres`, `content/stats/sanity_rules.tres`, testy `tests/test_death.gd`, `tests/test_remnant.gd`, `tests/test_sanity.gd`.

Opowieść

Kapsuła ma przy sobie 120 punktów doświadczenia i ani jednego wydanego. Przeciwnik testowy trafia raz, drugi, trzeci. Ekran gaśnie, na czarnym tle pojawia się czerwony napis „Nie żyjesz”.

Po chwili kapsuła stoi przy ognisku, cała i zdrowa, z zerem na liczniku doświadczenia. Tam, gdzie upadła, unosi się niebieskawa kula: „Odzyskaj pozostałość”. Doświadczenie wraca, ale coś w środku pęka. Poczytalność spada o 20.

Jeszcze jedna taka wyprawa i pasek poczytalności zjeżdża poniżej połowy. Świat się nie zmienia: te same ściany, to samo ognisko. Tylko światło robi się chore, tło ciemnieje, mgła gęstnieje. Kapsuła nie śpi. To jawa wygląda jak koszmar. Odpoczynek przy ognisku przywraca poczytalność i kolory.

A gdyby cios dosięgnął kapsułę w progu, gdy już przechodziła schodami do innego pokoju albo zasypiała przy ognisku? Śmierć poczeka, aż przejście się dokończy, i dopiero wtedy zabierze ją do ogniska. Kapsuła nigdy nie wstaje tam, gdzie upadła.

Gdyby kapsuła zginęła we śnie, obudzi się na jawie przy ognisku, a jej pozostałość zostanie we śnie. Odzyska ją tylko wtedy, gdy znów zaśnie (zwykłym snem, nie koszmarem) i odnajdzie to miejsce.

Pojęcie: stan zapisywany i stan liczony

> **Nie wszystko, co gra „pamięta”, trzeba zapisywać.**
>
> - **Pozostałość** to fakt z przeszłości: gdzie, kiedy i ile. Nie da się jej odtworzyć z innych danych, więc trafia do `GameState` i do zapisu gry.
> - **Koszmar na jawie** to wniosek z bieżącej poczytalności. Gdyby go zapisać osobno, mógłby się rozjechać z paskiem poczytalności (np. po zmianie progu przez autora). Dlatego `Stats` liczy go za każdym razem od nowa.
> - **Histereza** to dwa progi zamiast jednego: inny do wejścia, inny do wyjścia. Bez niej wygląd świata migotałby, gdy poczytalność drga wokół 50%.
>
> Zasada kciuka: **zapisuj fakty, wyliczaj wnioski.**

Klocek 1 [A]: strojenie śmierci i poczytalności

1. Otwórz `content/combat/death_rules.tres`. Ustaw, ile XP zostaje w pozostałości (**Remnant Xp Fraction**), ile zdrowia i poczytalności wraca po śmierci, ile poczytalności kosztuje odzyskanie (**Recovery Sanity Cost**), jak długo widać ekran śmierci.
2. Otwórz `content/stats/sanity_rules.tres`. **Break Fraction** to próg wejścia w koszmar na jawie, **Recover Fraction** to próg wyjścia (histereza: ustaw np. 0.6).
3. Wygląd koszmaru na jawie to pole **Nightmare Look** w węźle `RealmEnvironment` pokoju; ten sam, którego używa koszmar we śnie.
4. Sprawdź w grze: F9, „Postać”, ustaw poczytalność na 30.

Klocek 2 [B]: reakcja na pęknięcie poczytalności

Gotowy skrypt do węzła, który ma reagować na koszmar na jawie (np. dźwięk tła albo przeciwnicy w zadaniu 009):

```
extends Node

func _ready() -> void:
	var stats: Stats = Stats.find(get_tree())
	if stats != null:
		stats.sanity_broken_changed.connect(_on_sanity_broken_changed)
		_on_sanity_broken_changed(stats.is_sanity_broken())


func _on_sanity_broken_changed(broken: bool) -> void:
	print("Koszmar na jawie: ", broken)
```

Klocek 3 [C]: pozostałość przy śmierci

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

```
func leave_remnant() -> int:
	# 1. Każda śmierć unieważnia poprzednią pozostałość: druga szansa jest jedna.
	GameState.clear_remnant()
	var stats: Stats = Stats.find(get_tree())
	if stats == null:
		return 0
	# 2. Do pozostałości trafia ułamek niewydanego XP; reszta przepada.
	var fraction: float = rules.remnant_xp_fraction if rules != null else 1.0
	var amount: int = floori(stats.get_xp() * fraction)
	# 3. Gracz traci całe niewydane XP, także tę część, która nie trafiła do pozostałości.
	stats.set_value(Stats.XP, 0)
	# 4. Zapamiętujemy fakt: ile, w której rzeczywistości, w którym pokoju i gdzie.
	#    RemnantKeeper pokaże kulę tylko tam, gdzie zgadza się pokój i rzeczywistość.
	if amount > 0 and player != null:
		GameState.set_remnant(amount, GameState.get_realm(), LocationManager.get_current_room(),
player.global_position)
	return amount
```

Do przemyślenia: dlaczego pozostałość zapisujemy **przed** powrotem do ogniska, a nie po nim? Co by się stało z rzeczywistością i pokojem pozostałości, gdyby kolejność była odwrotna?

Sprawdź, że działa

Z katalogu głównego projektu, w PowerShellu:

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

Oczekiwane: w tabeli każdy wiersz ma `Wynik = OK`, w tym `test_death`, `test_remnant` i `test_sanity`. W grze: F9, „Postać”, +100 XP, potem „Walka”, „Zabij gracza”, wróć w miejsce śmierci i odzyskaj pozostałość.

Ciekawostka z kulis

Pierwsza wersja testów czekała na sygnał `respawned` (`await director.respawned`). Osobno test przechodził, ale w zbiorczym uruchomieniu menu deweloperskie zawiesiło się na tej linii: ekran śmierci trwał w teście 0.05 s, a zanim test doszedł do `await`, śmierć zdążyła się skończyć i sygnał już wybrzmiał. Na sygnał, który już był, czeka się w nieskończoność. Rozwiązanie: test odpytuje `DeathDirector.is_busy()` co klatkę, z limitem czasu. Zasada: **na sygnał czekaj tylko wtedy, gdy masz pewność, że jeszcze nie przyszedł.**

Druga pułapka wyszła przy przeglądzie: śmierć w trakcie przejścia portalem. `RealmManager.respawn_now` w trakcie przejścia odmawia i zwraca `false`, ale `DeathDirector` tego nie sprawdzał i od razu przywracał zdrowie. Kapsuła wstawała w miejscu, a chwilę później przejście kończyło się w pokoju docelowym portalu. Do tego śmierć i portal dzielą jedną nakładkę zaciemnienia. Nowe zaciemnienie przerywało stare, a przerwany tween nie wysyła `finished`, więc portal czekał na nie w nieskończoność. Teraz `DeathDirector` czeka na koniec przejścia, ponawia powrót do skutku i dopiero wtedy leczy, a `FadeOverlay` kończy przerwane czekanie. A gdyby przejście zawisło na dobre? Po 5 s `DeathDirector` zgłasza błąd i wymusza powrót. Czarny ekran bez wyjścia jest gorszy niż zgubione przejście. Zasada: **gdy funkcja może odmówić, sprawdź jej wynik, zanim zrobisz następny krok.** Opis: strona poprawki „Śmierć w trakcie przejścia”.

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

Revision #1
Created 2026-10-28 22:53:51 UTC by Tomasz Konieczny
Updated 2026-10-28 22:53:52 UTC by Tomasz Konieczny