Odcinek: Trzy pokoje i sen, który idzie za graczem

Odcinek: Trzy pokoje i sen, który idzie za graczem

> Szkic do BookStacka. Faza 1. Poziomy: **[A]** klikane w edytorze, **[B]** gotowy skrypt do wklejenia, **[C]** skrypt z komentarzami dydaktycznymi. Pliki w repozytorium: `content/main.tscn`, `core/game/game_root.gd`, `core/world/`, `content/world/rooms.tres`, `content/rooms/room_a.tscn`, `room_b.tscn`, `room_c.tscn`, `core/realm/realm_manager.gd`, testy `tests/test_location.gd`, `tests/test_quest_rooms.gd`, `tests/test_main_scene.gd`, `tests/test_realm_rooms.gd`.

Opowieść

Pokój testowy, w którym kapsuła spędziła pierwsze odcinki, zniknął. Na jego miejscu są trzy pokoje robocze.

W pokoju A leży klucz, płonie ognisko, a w północnej ścianie ciemnieje przejście. Kapsuła podnosi klucz i przechodzi: ekran gaśnie na chwilę, a ona stoi już w pokoju B, twarzą do wnętrza. Kamera przyszła razem z nią, bo kamera, gracz i menu nie należą do żadnego pokoju. Mieszkają w scenie głównej, a pokoje się pod nimi wymieniają.

W pokoju B są drzwi do piwnicy. Klucz je otwiera, za nimi jest przedsionek i kolejne przejście, do piwnicy C. Tam stoi drugie ognisko. Kapsuła zasypia przy nim, a gdy ekran się rozjaśnia, piwnica jest ciemniejsza i pełna mgły.

Kapsuła nie czeka. We śnie wraca do pokoju B: on też jest we śnie, bo sen idzie za graczem przez każde przejście. Pokój A we śnie ma inne ściany i czarny monolit w miejscu, gdzie na jawie stał stół.

Gdy przychodzi pora się obudzić, gra pamięta, gdzie gracz zasnął. Choć kapsuła stoi w pokoju A, obudzi się w piwnicy C, przy ognisku, przy którym zasnęła.

Pojęcie: scena główna i pokoje

> **Gra to jedna scena, która trzyma rzeczy trwałe, i wymieniane pod nią pokoje.**
>
> - **Scena główna** (`content/main.tscn`) ma gracza, kamerę, podpowiedź, menu pauzy i zaciemnienie. Te rzeczy nie znikają przy zmianie pokoju.
> - **Pokój** to osobna scena z podłogą, ścianami, światłem, obiektami i spawnami, bez gracza i kamery. `LocationManager` wstawia go pod węzeł `RoomHost`, a stary usuwa.
> - **Stan** (przedmioty, flagi, bieżący pokój, rzeczywistość, ostatnie ognisko) mieszka w `GameState`, więc pokój można zniszczyć i zbudować od nowa, a gra pamięta, że klucz jest zebrany, a drzwi otwarte.
> - **Rzeczywistość** też jest stanem, nie cechą pokoju. Nowy pokój przy starcie pyta `RealmManager`: „jawa, sen czy koszmar?” i ustawia swoje obiekty i wygląd.
>
> Zasada kciuka: **pokój jest wymienny, stan jest trwały.**

Klocek 1 [A]: nowy pokój i przejście do niego

1. Utwórz scenę `content/rooms/room_d.tscn` z korzeniem **Node3D**. Dodaj podłogę i ściany (np. **CSGCombiner3D** z **Use Collision** i **CSGBox3D** w środku), **WorldEnvironment**, **DirectionalLight3D** i węzeł `RealmEnvironment` z zasobami wyglądu (np. `content/realm/basic_*.tres`). Nie dodawaj gracza ani kamery.
2. Dodaj **Marker3D** ze skryptem `res://core/world/spawn_point.gd` i wpisz **Spawn Id**, np. `from_b`. Niebieska oś markera (−Z) to kierunek, w którym gracz będzie patrzył po wejściu.
3. Otwórz `content/world/rooms.tres` i dodaj do **Rooms** nowy **RoomInfo**: **Id** `room_d`, **Scene** nowy pokój, **Name Key** np. `ROOM_D_NAME` (dopisz go do `content/i18n/ui.csv`).
4. W pokoju, z którego ma prowadzić przejście, przeciągnij `res://core/world/portal.tscn` i ustaw **Target Room** `room_d` i **Target Spawn** `from_b`. Opcjonalnie **Required Flag**, jeśli przejście ma być zablokowane do czasu spełnienia warunku.

5. Przedmioty, które mają zostać zebrane na zawsze, dostają **Pickup Id** (np. `pickup_room_d_lamp`). Bez niego odradzają się przy każdym wejściu.

Klocek 2 [B]: powrót ze snu do pokoju ogniska

Gotowy kod w `res://core/realm/realm_manager.gd`. Ta funkcja woła się wewnątrz `return_to_waking()`, gdy ekran jest już zaciemniony:

```
func _load_last_point_room() -> void:
	var room_id: StringName = GameState.get_last_point_room()
	if room_id == &"" or not LocationManager.has_world():
		return
	if room_id != LocationManager.get_current_room():
		LocationManager.load_room_now(room_id)
```

Pokój ogniska zapisuje samo ognisko przy interakcji: `GameState.set_last_point(point_id, LocationManager.get_current_room())`.

Klocek 3 [C]: wymiana pokoju krok po kroku

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

```
func load_room_now(room_id: StringName, spawn_id: StringName = &"") -> bool:
	# 1. Czy taki pokój jest w rejestrze i czy scena główna się podłączyła?
	if not _can_load(room_id):
		return false
	var info: RoomInfo = _registry.find(room_id)
	var from: StringName = GameState.get_room()
	# 2. Ogłaszamy zmianę, zanim cokolwiek zniknie (np. dla dźwięku albo zapisu).
	room_changing.emit(from, room_id)
	if is_instance_valid(_room):
		# 3. Stary pokój wyjmujemy z drzewa OD RAZU. Samo queue_free() usunęłoby go
		#    dopiero na końcu klatki, a do tego czasu jego spawny i ogniska byłyby
```

```
        #    nadal w grupach i myliłyby się z nowymi.
        _room_host.remove_child(_room)
        _room.queue_free()
    # 4. Nowy pokój: instancja sceny pod RoomHost. Jego _ready() zrobi resztę,
    #    np. RealmAware ustawi się według bieżącej rzeczywistości, a Pickup
    #    z zapisanym zebraniem sam zniknie.
    _room = info.scene.instantiate()
    _room.name = String(room_id)
    _room_host.add_child(_room)
    # 5. Stan: bieżący pokój trafia do GameState, więc też do zapisu gry.
    GameState.set_room(room_id)
    # 6. Gracz staje na spawnie (pozycja i obrót markera, zerowa prędkość).
    _place_player(spawn_id)
    room_changed.emit(room_id)
    return true
```

Do przemyślenia: zapis gry pamięta pokój, ale nie pozycję gracza w pokoju, więc po wczytaniu gracz stoi na pierwszym spawnie. Kiedy to wystarcza, a kiedy trzeba zapisać dokładną pozycję?

Sprawdź, że działa

Z katalogu głównego projektu, w PowerShellu:

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

Oczekiwane: w tabeli każdy wiersz ma `Wynik = OK`, w tym `test_location`, `test_quest_rooms`, `test_main_scene` i `test_realm_rooms`. W edytorze: F5 uruchamia scenę główną i pokój A.

Ciekawostka z kulis

Usuwając pokój testowy, nie można było po prostu skasować jego testu. Każde sprawdzenie z `test_room.gd` musiało znaleźć nowy dom: okablowanie kamery i pauzy trafiło do `test_main_scene.gd`, klucz i drzwi z podpowiedziami do `test_quest_rooms.gd`, a wersja snu pokoju razem ze ścianami, stołem i monolitem przeszła do pokoju A i jest sprawdzana tak samo jak wcześniej.

Jeden z nowych testów na początku „padał”: po obudzeniu z koszmaru gracz lądował w piwnicy, a test spodziewał się pokoju A. Kod miał rację, bo ostatnim ogniskiem było to w piwnicy. Pomylił się test. To dobry przykład, że czerwony test najpierw się czyta, a dopiero potem poprawia kod.

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

Revision #2
Created 2026-10-28 00:48:06 UTC by Tomasz Konieczny
Updated 2026-10-28 22:53:47 UTC by Tomasz Konieczny