Odcinek: Drzwi, które tylko udawały, że się otwierają

Odcinek: Drzwi, które tylko udawały, że się otwierają

> Szkic do BookStacka. Faza 1. Poziomy: **[A]** klikane w edytorze, **[B]** gotowy skrypt do wklejenia, **[C]** skrypt z komentarzami dydaktycznymi. Pliki w repozytorium: `core/interaction/door.tscn`, `core/interaction/door.gd`, `core/camera/camera_control.gd`, testy `tests/test_door_passage.gd`, `tests/test_camera_control.gd`.

Opowieść

Klucz do piwnicy jest w kieszeni. Kapsuła podchodzi do drzwi, na ekranie pojawia się **„Otwórz”**, skrzydło obraca się płynnie na zawiasie i staje otworem. Kapsuła rusza naprzód i… zatrzymuje się w progu, jakby stała przed szybą.

Drzwi były otwarte tylko dla oka. Dla fizyki wciąż stały zamknięte: niewidzialna płyta w dokładnie tym miejscu, gdzie przed chwilą było skrzydło. Testy z poprzedniego odcinka tego nie wyłapały, bo sprawdzały kąt obrotu, a nie to, czy da się przejść.

Tym razem zaczęliśmy od testu, który robi to co gracz: prowadzi kapsułę przez otwór. Na starych drzwiach test od razu pokazał błąd: prześwit 0 m, przejście zablokowane na każdym torze. Po poprawce ta sama kapsuła przechodzi na drugą stronę ze środka i z obu krawędzi otworu.

Przy okazji gracz dostał to, czego brakowało od pierwszego pokoju: **może się rozejrzeć**. Ruch myszy albo prawej gałki obraca kamerę wokół kapsuły. Gdy jednak gracz wejdzie w róg pokoju i obraz przejmie kamera stałego ujęcia, obracanie się wyłącza, bo tamto ujęcie ma zostać takie, jakie

zaplanował autor.

Pojęcie: kolizja a obrót rodzica

> **W Godocie mesh i kolizja to dwa osobne światy: obraz i fizyka.**
>
> - Zwykłe węzły 3D (Node3D, MeshInstance3D) poruszają się razem z rodzicem. Obracasz rodzica, a obraz dzieci obraca się z nim.
> - Ciało fizyczne (StaticBody3D, AnimatableBody3D, CharacterBody3D) żyje dodatkowo w silniku fizyki. Silnik dowiaduje się o jego ruchu wtedy, gdy porusza się **samo ciało**.
> - `AnimatableBody3D` z `sync_to_physics = true` przenosi do fizyki własny ruch, np. z tweena czy animacji. Ruchu węzła nad nim nie przenosi.
>
> Wniosek: **obracaj ciało, nie jego rodzica.** Jeśli coś ma się obracać wokół zawiasu, niech zawiasem będzie samo ciało, a kształt kolizji i mesh przesuń w jego dzieciach.
>
> I druga lekcja: **test ma robić to, co gracz.** Kąt obrotu był poprawny, a przejście i tak zablokowane.

Klocek 1 [A]: drzwi z prawidłowym zawiasem

1. Otwórz `core/interaction/door.tscn`.
2. Zawiasem jest węzeł **Hinge** typu `AnimatableBody3D`. W Inspektorze ma zaznaczone **Sync To Physics**. Nie odznaczaj tego pola.
3. Pod zawiasem są **CollisionShape3D** (pudełko 1.2 × 2.2 × 0.1) i **Mesh**. Oba są przesunięte o 0.6 m w osi X, czyli o pół szerokości skrzydła.
4. Chcesz drzwi innej szerokości? Zmień rozmiar pudełka i mesha, a przesunięcie dzieci ustaw na połowę nowej szerokości. Węzeł **Hinge** ustaw na krawędzi otworu.
5. Otwór w ścianie musi być co najmniej tak szeroki jak skrzydło. W pokoju testowym ma 1.2 × 2.2 m, a kapsuła gracza ma 0.7 × 1.7 m.

Klocek 2 [B]: kamera sterowana myszą i padem

Gotowy plik: `res://core/camera/camera_control.gd`. Wystarczy go podpiąć w scenie:

1. Dodaj do pokoju węzeł **Node** i nazwij go `CameraControl`.
2. Przypnij skrypt `res://core/camera/camera_control.gd`.
3. W Inspektorze wskaż **Player Camera** (kamera za plecami) i **Host** (`MainCamera3D/PhantomCameraHost`).
4. Dostosuj czułość (**Mouse Sensitivity**, **Stick Sensitivity**), granice pochylenia (**Pitch Min**, **Pitch Max**) i ewentualnie **Invert Y**.

Tak wygląda serce komponentu:

```
func apply_look(delta_degrees: Vector2) -> bool:
	if not is_input_enabled():
		return false
	var pitch_delta: float = -delta_degrees.y if invert_y else delta_degrees.y
	yaw = wrapf(yaw + delta_degrees.x, -180.0, 180.0)
	pitch = clampf(pitch + pitch_delta, pitch_min, pitch_max)
	_apply_to_camera()
	return true
```

W grze: Esc zwalnia mysz, kliknięcie w okno przechwytuje ją z powrotem.

Klocek 3 [C]: test, który chodzi jak gracz

Wersja z komentarzami dydaktycznymi. Pełny kod jest w `res://tests/test_door_passage.gd`.

```
## Prowadzi kapsułę z START_X wzdłuż +X przez WALK_FRAMES klatek fizyki. Zwraca końcowe x.
func _walk(lane: float) -> float:
	# 1. Stawiamy kapsułę przed drzwiami, na wybranym torze (środek albo krawędź otworu).
	_walker.global_position = Vector3(START_X, 0.05, lane)
	_walker.velocity = Vector3.ZERO
	await _physics_frames(1)
```

```
	for i: int in WALK_FRAMES:
		# 2. Czekamy na klatkę fizyki: kolizje liczą się w jej rytmie, nie w rytmie obrazu.
		await physics_frame
		var delta: float = 1.0 / Engine.physics_ticks_per_second
		# 3. Idziemy prosto przed siebie, z grawitacją, jak gracz trzymający gałkę w przód.
		_walker.velocity.x = SPEED
		_walker.velocity.z = 0.0
		_walker.velocity.y = 0.0 if _walker.is_on_floor() else _walker.velocity.y - 9.8 * delta
		# 4. move_and_slide to ten sam ruch co u gracza: zatrzyma się na ścianie albo skrzydle.
		_walker.move_and_slide()
		if _walker.global_position.x > PASSED_X + 0.5:
			break
	# 5. Wynik to miejsce, w którym kapsuła skończyła: przed drzwiami albo za nimi.
	return _walker.global_position.x
```

Do przemyślenia: test mierzy też prześwit promieniami puszczanymi wzdłuż przejścia. Dlaczego promienie puszczone **w poprzek** otworu nie wykryłyby zamkniętego skrzydła? Podpowiedź: skąd startowałby taki promień?

Sprawdź, że działa

Z katalogu głównego projektu, w PowerShellu:

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

Oczekiwane: w tabeli każdy wiersz ma `Wynik = OK`, także `test_door_passage` i `test_camera_control`.

Ciekawostka z kulis

Na pierwszej wersji testu kamery wszystkie obroty „nie działały”. Okazało się, że host Phantom Camera wybiera aktywną kamerę dopiero po pierwszej klatce **procesu**, a test czekał tylko na klatki **fizyki**. Do tego czasu `CameraControl` uznawał, że aktywna jest „jakaś inna kamera”, i grzecznie ignorował wejście. Pomogło czekanie na oba rodzaje klatek. Ta pułapka jest opisana na stronie referencyjnej kamery.

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

Revision #5
Created 2026-10-27 21:45:45 UTC by Tomasz Konieczny
Updated 2026-10-28 22:53:45 UTC by Tomasz Konieczny