Odcinek: Ognisko, które leczy, uczy i pamięta

Odcinek: Ognisko, które leczy, uczy i pamięta

> Szkic do BookStacka. Faza 1. Poziomy: **[A]** klikane w edytorze, **[B]** gotowy skrypt do wklejenia, **[C]** skrypt z komentarzami dydaktycznymi. Pliki w repozytorium: `core/stats/`, `core/save/save_manager.gd`, `core/ui/level_up_screen.tscn`, `core/ui/save_slots_screen.tscn`, `content/stats/`, testy `tests/test_stats.gd`, `tests/test_leveling.gd`, `tests/test_save.gd`.

Opowieść

Kapsuła ma teraz zdrowie i poczytalność. Na razie nic jej nie rani, ale menu deweloperskie potrafi ją „podrapać”: zdrowie spada do 40, poczytalność do 60.

Przy ognisku menu ma trzy nowe pozycje. **„Odpocznij”** przesuwa zegar i przywraca zdrowie oraz poczytalność. **„Zaśnij”** zabiera do snu, ale ran nie leczy: sen to nie odpoczynek. **„Awans”** otwiera ekran z doświadczeniem i opcjami: Wytrzymałość, Wola, Opanowanie. Każdy zakup zjada trochę doświadczenia, podnosi poziom o jeden i powiększa jedną z pul. Następny zakup kosztuje więcej.

W menu pauzy, z dala od ognia, jest pozycja **„Postać”**. Ten sam ekran, ale Wytrzymałość i Wola są wyszarzone z dopiskiem „Tylko przy ognisku”. Opanowanie da się kupić wszędzie.

Ognisko daje jeszcze jedno: **„Zapisz”**. Trzy sloty, każdy z opisem: gdzie, na jakim poziomie, który dzień i która godzina. Po wczytaniu, nawet po wyjściu z gry, kapsuła stoi dokładnie tam, gdzie stała, z tymi samymi ranami i tym samym doświadczeniem.

Pojęcie: stan, który przeżywa wyłączenie gry

> **Zapis gry to zdjęcie stanu, a nie zdjęcie sceny.**
>
> - Gra nie zapisuje węzłów, drzew ani kamer. Zapisuje dane, z których scenę da się odbudować: `GameState` (przedmioty, flagi, czas, sen, statystyki) oraz pokój i pozycję gracza.
> - Dlatego od początku projektu wszystko, co ma przetrwać, mieszka w `GameState`, a statystyki w węźle `Stats` są tylko logiką nad tymi danymi.
> - **Wersja** w pliku pozwala wczytać stare zapisy po zmianach w grze: brakujące pola dostają wartości domyślne. Plik z nowszej wersji jest odrzucany z czytelnym komunikatem.
> - **Atomowość**: nowy zapis powstaje obok starego (plik `.tmp`) i podmienia go dopiero, gdy jest cały. Wyłączony prąd w połowie zapisu nie zniszczy poprzedniego.
>
> Zasada kciuka: **najpierw sprawdź, potem zmień.** Uszkodzony plik nie może zepsuć gry, w którą właśnie grasz.

Klocek 1 [A]: nowa opcja awansu

1. W `content/stats/` utwórz zasób **LevelUpOption** (prawy klik → *Nowy zasób* → `LevelUpOption`), np. `levelup_focus.tres`.
2. Ustaw **Id** (`focus`), **Name Key** (`LEVELUP_FOCUS_NAME`; dopisz go do `content/i18n/ui.csv`), **Target Stat** (np. `stat_sanity.tres`), **Amount**, **Cost Multiplier**.
3. Zaznacz **Requires Meditation**, jeśli opcja ma być dostępna tylko przy ognisku.
4. Otwórz `content/stats/level_up_table.tres` i dopisz opcję do listy **Options**.
5. Uruchom grę, naciśnij F9, w sekcji „Postać” dodaj sobie XP i obejrzyj ekran awansu przy ognisku.

Inne liczby (zdrowie startowe, przyrost na poziom, koszty kolejnych poziomów) zmienia się w `content/stats/stat_*.tres` i `level_curve.tres`, a leczenie przy konkretnym ognisku w polu **Rest Rules** (`MeditationRules`).

Klocek 2 [B]: zapis atomowy

Gotowy kod w `res://core/save/save_manager.gd`:

```
var temp_path: String = path + ".tmp"
var file: FileAccess = FileAccess.open(temp_path, FileAccess.WRITE)
if file == null:
	return _fail(slot, ERROR_WRITE, "%s: %s" % [temp_path,
error_string(FileAccess.get_open_error())])
file.store_string(JSON.stringify(data, "\t"))
file.close()
var error: Error = DirAccess.rename_absolute(temp_path, path)
```

Klocek 3 [C]: zakup awansu

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

```
func level_up(option: LevelUpOption, in_meditation: bool) -> bool:
	# 1. Czy wolno? Przy ognisku (requires_meditation), poniżej najwyższego
	#    poziomu i z wystarczającym doświadczeniem.
	if not can_level_up(option, in_meditation):
		return false
	# 2. Płacimy: koszt z krzywej poziomów razy mnożnik opcji.
	var cost: int = level_up_cost(option)
	set_value(XP, get_xp() - cost)
	# 3. Poziom o jeden w górę. Następny awans będzie droższy.
	GameState.set_stat(LEVEL, get_level() + 1)
	value_changed.emit(LEVEL, float(get_level()))
	# 4. Premia do maksimum statystyki trafia do GameState, więc też do zapisu gry.
	var target: StringName = option.target_stat.id
	GameState.add_stat_bonus(target, option.amount)
	# 5. Pula (zdrowie, poczytalność) rośnie razem z maksimum, żeby awans było czuć od razu.
	if option.target_stat.is_pool:
		change(target, option.amount)
	else:
		value_changed.emit(target, get_value(target))
	leveled_up.emit(get_level())
	return true
```

Do przemyślenia: dlaczego `add_xp()` nie podnosi poziomu sama? Co zmieniłoby to w rytmie gry, w której gracz wraca do ogniska, żeby „wydać” to, co zebrał?

Sprawdź, że działa

Z katalogu głównego projektu, w PowerShellu:

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

Oczekiwane: w tabeli każdy wiersz ma `Wynik = OK`, w tym `test_stats`, `test_leveling` i `test_save`. W grze: F9, „Postać”, +1000 XP, potem ognisko, „Awans”, „Zapisz”, a na koniec Esc, „Wczytaj”.

Ciekawostka z kulis

Test uszkodzonego zapisu przechodził, a i tak by „padł” w zbiorczym uruchomieniu. Funkcja `JSON.parse_string()` przy złym pliku wypisuje do logu `ERROR: Parse JSON failed`, a skrypt testów słusznie traktuje każdą linię `ERROR:` jako błąd. Rozwiązanie: `JSON.new().parse()`, które zwraca kod błędu (i numer linii do komunikatu), nie krzycząc w logu. Gracz z uszkodzonym plikiem zobaczy spokojne „Plik zapisu jest uszkodzony”, a nie czerwony błąd silnika.

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

Revision #1
Created 2026-10-28 22:53:50 UTC by Tomasz Konieczny
Updated 2026-10-28 22:53:50 UTC by Tomasz Konieczny