Słownik młodego GITa

Z pomocnymi analogiami!

1. Co jest czym

Pojęcie (po ang.)	Wersja polska	Analogia
Repository	Repozytorium	Lodóweczka.
Clone	Klonowanie	Pobranie gry ze Steam'a (wraz z kodem źródłowym) po raz pierwszy.
Commit	Zatwierdzenie	Odpowiednie zapieczętowanie garnka.
Push	Wypchnięcie	Włożenie dania do wspólnej lodówki.
Fetch	Pobranie info	Sprawdzenie, czy jakieś nowe frykasy nie pojawiły się w lodówce.
Pull	Pobranie (i scalenie)	Wyjęcie rzeczy z lodówy.
Branch	Gałąź	Alternatywna rzeczywistość (Kopia świata), gdzie możesz bezpiecznie psuć.
Merge	Scalanie	Moment, gdy Twoja alternatywna rzeczywistość łączy się z główną grą.
Conflict	Konflikt	Dwie osoby chcą postawić garnek w tym samym miejscu w lodówce. Fizyka na to nie pozwala.

Warto jeszcze zanotować co to jest **Directory**. Jest to **katalog** lub **folder**, gdzie trzymacie cały projekt.

2. Jak nazywać Branche? (Naming Convention)

Dobra nazwa mówi, co jest w środku bez otwierania.

Używaj przedrostków i myślników:

- `feature/` – Nowa rzecz (np. `feature/inventory-system`)
- `bugfix/` – Naprawa błędu (np. `bugfix/player-jump`)
- `hotfix/` – Pilna naprawa, gdy gra nie działa (np. `hotfix/crash-on-start`)
- `design/` – Grafiki, UI, Wygląd (np. `design/main-menu-bg`)
- `refactor/` – Sprzątanie kodu (np. `refactor/cleanup-scripts`)

Złe nazwy: `poprawki`, `moje-zmiany`, `test`, `Jacek_Feature`
Dobre nazwy: `feature/enemy-ai`, `design/update-logo`

3. Scenariusze ratunkowe (Co robić, gdy...)

Robię grafiki / dźwięki

Sytuacja: Zrobiłeś model (`sword.fbx`) i teksturę (`sword.png`).

1. Wrzucasz pliki do folderu w projekcie.
2. W GitHub Desktop widzisz swoje pliki.
3. **BARDZO WAŻNE:** Korzystając z Godota, czasem pojawi się plik `.import`. **Musisz je zcommitować razem z grafiką!** Bez nich silnik u kolegi zwariuje (ale lekko).
4. Commit -> Push.
5. Daj znać zespołowi: "Assety na repo".

Koduję lub naprawiam błędy

Sytuacja: Postać przenika przez ściany.

1. GitHub Desktop -> **Current Branch** -> **New Branch**.
2. Nazwa: `bugfix/wall-collision`.
3. Naprawiasz kod.
4. Wracasz do GitHub Desktop -> Sprawdzasz zmiany (czerwone/zielone linie).
5. Commit: `Fix collision layer check`.
6. **Publish Branch** (Push).

CZERWONY ALERT (Merge Conflict)

Sytuacja: Klikasz **Pull**, a GitHub Desktop krzyczy, że jest **Conflict**.

1. **Nie panikuj!** Nikt nikogo nie zabił! Damy radę razem.
2. **GitHub Desktop** zapyta Cię, co zrobić z plikiem, który powoduje problem.
3. Masz zazwyczaj opcje (często trzeba kliknąć w "Open in..." lub wybrać opcję w okienku):
   - **Accept Incoming Change:** "Wersja kolegi jest ważniejsza/nowsza. Bierzemy jego."
   - **Accept Current Change:** "Ja mam rację, moja wersja zostaje."
   - **Manual Merge:** "Musimy to połączyć ręcznie w edytorze tekstu".
4. Po wybraniu -> Klikasz **Commit Merge**.
5. Robisz **Push**. Kryzys zażegnany.

"Klops się zrobił, chcę cofnąć!"

Sytuacja: Przez godzinę pisałeś kod, gra przestała działać, chcesz wrócić do stanu z rana.

1. W **GitHub Desktop** na liście "Changes" zaznacz pliki, które zepsułeś.
2. Kliknij Prawy Przycisk Myszy na daną zmianę lub `x changed files` -> **Discard changes**.
   - x = liczba zaznaczonych zmian.
3. Potwierdź.
4. **Efekt:** Twoja *niezapisana* (niezcommitowana) praca znika, pliki wracają do ostatniego zapisu.