KOMPLETNÁ DOKUMENTÁCIA K JEDNOTLIVÝM SKRIPTOM V PROJEKTE, ROZDELENÁ DO LOGICKÝCH SKUPÍN PODĽA ICH FUNKCIE A VZÁJOMNÝCH VZŤAHOV

---

1. ZÁKLADNÉ DÁTOVÉ ŠTRUKTÚRY A SYSTÉMY HRY

BuildingData.cs

Účel:
Uchováva údaje o budove (napr. názov, popis, základný a maximálny výkon, cena, environmentálne náklady, údržba).

Kľúčové vlastnosti:

- **id, displayName, tagClass, description** – Identifikátor, zobrazovaný názov, trieda (napr. „Wind“, „Hydro“), popis.
- **basePower, maxPower** – Základný a maximálny výkon.
- **connectionRange** – Maximálny dosah spojenia (napr. pre káble).
- **cost, ecoCost, maintenanceCost** – Cena na postavenie, ekologická daň a cena údržby.

Vzťahy / Použitie:

- Skript **BuildingInfoLibrary** používa BuildingData pre mapovanie a vyhľadávanie.
- Ďalšie triedy (napr. ConnectableObject) z neho čerpajú informácie o produkcii, cene atď.

BuildingInfoLibrary.cs

Účel:
Uchováva slovník (dictionary) „ID budovy -> BuildingData“, aby sa skripty vedeli dostať k parametrom danej budovy podľa reťazcového ID.

Kľúčové vlastnosti a metódy:

- buildingMap (Dictionary) – Statická mapovacia tabuľka.
- Get(string id) – Vráti príslušný BuildingData alebo null, ak ID neexistuje.

Vzťahy / Použitie:

- Každý ConnectableObject si z tohto skriptu dokáže načítať svoj BuildingData (podľa buildingID).

ConnectableObject.cs

Účel:
„Základ“ pre budovu, ktorá sa môže pripájať do elektrickej siete.
Obsahuje logiku ohľadom:

- pripojení (káble),
- produkcie výkonu (powerProduction),
- parametrov budovy (načítaných z BuildingData podľa buildingID).

Kľúčové vlastnosti:

- **buildingID** – String, na základe ktorého načíta BuildingData.
- **connectionPoint** – Transform, bod pripojenia pre káble.
- **powerProduction** – Aktuálny (runtime) výkon budovy.
- **connections** – Zoznam naviazaných objektov (iných ConnectableObject) a ich referencií na ConnectionCable.
- **isInNetwork** – Indikátor, či je objekt v sieti (napr. BFS dobehlo až k nemu).

Hlavné metódy:

- AddConnection(other, cable), RemoveConnection(other) – Správa pripojení na iné objekty a udržovanie referencií na kábloch.
- SetInNetwork(bool state) – Nastaví, či je budova v sieti.
- ShowConnectionRangeIndicator(bool enable) – Zapína/vypína pomocný kruh (RangeIndicator) zobrazujúci dosah pripojenia.

Vzťahy / Použitie:

- Používa **BuildingInfoLibrary** na načítanie hodnôt z BuildingData.
- Mnoho špecializovaných budov (napr. WindTurbine, SolarPanelStatic, BioBurner, atď.) obsahuje tento komponent.
- Pri vytváraní spojení (kábla) hrá hlavnú úlohu (každé pripojenie je ConnectableObject – ConnectableObject).

NetworkEvents.cs

Účel:
Jednoduchý systém eventov ohľadom zmien v sieti (OnNetworkChanged).
Umožňuje spustiť recalculations sietí, napr. keď sa pripojí či odpojí budova.

Vzťahy / Použitie:

- **NetworkManager** (PowerNetworkManager) a iné skripty (napr. ConnectableObject, ConnectionCable) volajú TriggerNetworkChanged() ak došlo k zmene v prepojení.
- MilestoneController môže taktiež odoberať event, aby mohol refreshnúť niektoré UI a pod.

NetworkManager.cs (obsah triedy: PowerNetworkManager)

Účel:
Spravuje výpočty pripojenia (využitím BFS) – zisťuje, ktoré uzly v sieti sú pripojené k goal (hlavnému odberateľovi).
Kontroluje, či je výkon dostatočný oproti nejakému requiredPower.

Kľúčové metódy:

- RecalculateNetwork() – Vykoná BFS od goal a označí všetky budovy vo vetve ako isInNetwork = true. Ostatné označí ako odpojené. Zároveň spočíta sumu powerProduction pripojených uzlov.

Vzťahy / Použitie:

- Odoberá sa na event NetworkEvents.OnNetworkChanged.
- Kedykoľvek sa sieť zmení (pridá kabeláž, budovu, atď.), zavolá sa BFS.

EcoBalanceManager.cs

Účel:
Spravuje ekologickú rovnováhu (ecoBalance).

- Rieši odstraňovanie vegetácie (stromov, detailov) pri umiestnení budovy a generuje zmeny v ecoBalance.
- Rieši aj „pollution effect“ (postupné „odumieranie“ stromov).

Kľúčové vlastnosti:

- **ecoBalance** – Aktuálna eko-hodnota (napr. 1000).
- RemoveVegetationUnderBuilding(...) – Zistí stromy a detaily v oblasti budovy a odstráni ich, zníži ecoBalance.
- ApplyPollutionEffectGradually(...) – Postupne mení stromy na mŕtve v určitom okruhu okolo budovy (s random šancou).

Vzťahy / Použitie:

- Volané v BuildingPlacer (po definitívnom umiestnení budovy).

- Spolupracuje s UIManager (ukazuje vyskakovacie texty, animáciu zmeny eko-balansu).

CreditsManager.cs

Účel:
Spravuje aktuálne kredity hráča.
Poskytuje metódy na míňanie a pridávanie kreditov (TrySpend, AddCredits).
Volá metódy z UIManager na aktualizáciu UI a vyskakovacie notifikácie o + / - kreditov.

Vzťahy / Použitie:

- BuildingPlacer používa TrySpend pri stavaní budovy.
- MilestoneController používa AddCredits pri odmeňovaní.

MilestoneController.cs (obsah triedy: MilestoneUIController)

Účel:
Rieši „milníky“ (stupne rozvoja). Má slider, text a ďalšie UI prvky.

- Sleduje celkový výkon siete (pomocou BFS cez NetworkEvents) a keď presiahne requiredPower, zobrazí tlačidlo na upgrade.
- Pri upgrade zvyšuje hráčovi kredity (vzorec), mení parametre (napr. requiredPower pre ďalší milestone) a teraz navyše vie spúšťať aj **milestone udalosti**.

Kľúčové vlastnosti:

- powerSlider, powerText – UI elementy pre zobrazenie aktuálneho a požadovaného výkonu.
- requiredPower – Požadovaný výkon na dosiahnutie milníka.

Vylepšenia (aktualizované):

- Po kliknutí na „Upgrade“ sa navyše vyberie náhodná udalosť z MilestoneEventLibrary.
- Tieto efekty (napr. bonus k veterným, solárnym či bio budovám) sa aplikujú na vybrané budovy cez mechanizmus pendingCount a synchronizáciu s MilestoneEvents.
- Po tom, čo budovy dokončia aktualizáciu (notifikované cez MilestoneEvents), sa vypočíta finálna odmena (zahŕňa aj prípadné bonusy).

Metódy (zhrnuté):

- OnUpgradeButtonPressed() – Zvyšuje milníkovú úroveň, pridelí kredity, spustí (náhodnú) milestone udalosť a následne zdvihne requiredPower a refreshne UI.
- Ďalšie interné metódy pre prácu s UI a volanie MilestoneEventManager.

Vzťahy / Použitie:

- Počúva na NetworkEvents.OnNetworkChanged => RefreshPower().
- Zisťuje sumu výkonu (cez FindObjectsByType<ConnectableObject>) a porovnáva s requiredPower.
- Pri upgrade spolupracuje s **MilestoneEventManager**, **MilestoneEvents**, **MilestoneEventLibrary** na aplikácii udalosti.

GameManager.cs (obsah triedy: GameManager)

Účel:
Singleton, ktorý prežíva medzi scénami.
Obsahuje funkcie na reštart scény (RestartGame()) alebo úplné vypnutie (QuitGame()).

Vzťahy / Použitie:

- Dá sa volať z UI, keď hráč stlačí napr. „Reštart“ alebo „Quit“.

MilestoneEventManager.cs

Účel:

- Riadi účinné milestone udalosti („veterno“, „slnko“, „sucho“ a pod.).
- Globálne ich aplikuje na všetky ConnectableObject, ktoré majú zhodný tagClass.

Metódy:

- ApplyMilestoneEvent(MilestoneEventData) → Aktualizuje výkon všetkých budov podľa danej udalosti.
- ClearMilestoneEvent() → Odstráni vplyv predchádzajúcej udalosti.
- GetActiveEvent() → Vráti aktuálnu (aktuálne aplikovanú) udalosť.

MilestoneEvents.cs

Účel:

- Poskytuje udalosti pre synchronizáciu po zmene milestone efektov.

Udalosti:

- OnMilestoneChanged → Spustené, keď sa aplikujú efekty (vyvolá recalculáciu).

- OnMilestoneRefreshReady → Spustené, keď všetky dotknuté budovy dokončia aktualizáciu.

Logika:

- TriggerMilestoneChanged(int countNeeded) → Nastaví počet budov, ktoré ešte musia zavolať NotifyDynamicBuildingUpdated().
- NotifyDynamicBuildingUpdated() → Zníži counter; ak dosiahne nulu, spustí OnMilestoneRefreshReady.

MilestoneEventLibrary.cs

Účel:
Ukladá preddefinované milestone udalosti (slovník eventMap), napr.:

- "windy" → Wind +20%
- "solar_flare" → Solar +30%
- "drought" → Hydro a Bio -30%
- "overcast" → Solar -40%

Metódy:

- Get(string id)
- GetAll()
- GetRandomEvent()

Poznámka:
Automaticky sa nahrávajú ikony z Resources/MilestoneIcons/{id} (podľa id udalosti).

DataTypes.cs *(rozšírené o MilestoneEventData)*

Trieda MilestoneEventData

- **id, description** → Identifikátor a popis udalosti.
- **targetBuildingClass** → Ktoré „tagy“ budov to ovplyvňuje (napr. „Wind“).
- **outputMultiplier, maxMultiplier** → Násobí produkciu a maximálnu kapacitu.
- **flatOutputBonus, flatMaxBonus** → Pridáva pevné čísla k produkcii / max kapacite.
- **icon** → Sprite (grafika) pre UI.
- Affects(classTag) → Pomocná metóda na filtrovanie (vráti true, ak classTag spadá pod udalosť).

---

2. SYSTÉM STAVBY BUDOV A PRIPOJENÍ (PLACEMENT + KÁBLE)

BuildingPlacer.cs

Účel:
Rieši logiku umiestňovania budov do scény myšou („Ghost“ preview, valid/invalid sfarbenie, kontrola kolízií a terénu).
Po potvrdení (ľavé kliknutie) reálne budovu umiestni, odráta kredity, spustí environmentálne efekty.

Kľúčové črty:

- isPlacing – Indikátor, či je hráč v procese klikania a hľadania miesta.
- currentPreview – Inštancia prefabu, zobrazuje sa ako ghost.
- IsValidPlacement() – Kontroluje sklon terénu, kolízie, vzdialenosť, atď.
- ConfirmPlacementRoutine() – Spustí animáciu „dopadnutia“, odpočíta kredity (cez CreditsManager), zavolá EcoBalanceManager atď.
- isHydroDamPlacement – Rozlišuje, či ide o normálnu budovu alebo špecifický HydroDam.

Vzťahy / Použitie:

- Používa CreditsManager na kontrolu stavu kreditov.
- Po finalizácii vyvolá EcoBalanceManager.RemoveVegetationUnderBuilding(...) a ApplyPollutionEffectGradually(...).

ConectionManager.cs (obsah triedy: ConnectionManager)

Účel:
Rieši vytváranie a mazanie elektrických káblov medzi budovami (ConnectableObject).
Má tri režimy: 0 = none, 1 = build, 2 = delete.

V režime build:

- Prvý klik – vyberie budovu.
- Druhý klik – ak kliknem na inú budovu a je v dosahu, vytvorí sa ConnectionCable.

V režime delete:

- Ak kliknem na existujúci kábel, vymaže ho (zničí aj jeho referenciu v pripojených budovách).

Kľúčové vlastnosti:

- cablePrefab – Prefab kábla.

- selected – Aktuálne vybratá budova (pri 1. kliku).
- tempCable – Preview kábel pri ťahaní z jednej budovy k druhej.

Vzťahy / Použitie:

- Interaguje s ConnectableObject (pridáva/odoberá prepojenia).
- Po vytvorení alebo zmazaní kábla volá NetworkEvents.TriggerNetworkChanged().

ConnectionCable.cs

Účel:
Samotný „kábel“ medzi dvoma ConnectableObject.
Rieši kreslenie krivky (LineRenderer), kolíziu s prekážkami, vymazanie.

Kľúčové vlastnosti:

- lr (LineRenderer) – zobrazuje zakrivený tvar kábla (Bézierova krivka).
- buildingA, buildingB – dve ConnectableObject, ktoré spája.
- DeleteConnection() – pri mazaní kábla volá RemoveConnection na oboch budovách a Destroy(this).

Vzťahy / Použitie:

- Vzniká a je spravovaný triedou ConnectionManager.

CableUIButton.cs

Účel:
Jednoduchý skript na UI tlačidlo pre zmenu mode v ConnectionManager (napr. 1 = build, 2 = delete).
V Update() mení farbu buttonu (normalColor / selectedColor) podľa toho, či je v móde.

Vzťahy / Použitie:

- Patrí k UI, priamo referencuje ConnectionManager (singleton Instance).

---

3. ŠPECIALIZOVANÉ BUDOVY

WindTurbine.cs

Účel:
Budova, ktorá dynamicky mení powerProduction podľa výšky terénu a testu vzdialenosti od iných turbín (proximityConflict).

ComputeHeightMultiplier() vyhodnocuje, koľko vzoriek je v malej / strednej / vysokej nadmorskej výške.

Vzťahy / Použitie:

- Vlastní ConnectableObject, do ktorého zapisuje výsledný powerProduction.
- Používa RangeIndicator ako tzv. „wind range“.

SolarPanelStatic.cs

Účel:
Stacionárny solárny panel. Výkon závisí od uhla voči slnku (RenderSettings.sun) a od obštrukcií (raycast).
Počítaný multiplikátor: dot(sunDirection, panelUp) + preverenie, či nie je lúč blokovaný.

Vzťahy / Použitie:

- Podobne ako WindTurbine, upravuje powerProduction v pripojenom ConnectableObject.
- Po finalizácii zruší ghost režim.

BioBurner.cs

Účel:
Výkon je závislý od počtu neobsadených fariem v určitom okruhu.
V ghost preview móde počíta, koľko by bolo „Farm“ k dispozícii, a z toho vyvodí dočasný výkon.
Po finalizácii si tie farmy rezervuje, aby iný BioBurner nemohol použiť tie isté.

Vzťahy / Použitie:

- Komunikuje s komponentom Farm (rezervuje / uvoľní farmy).
- Taktiež využíva ConnectableObject a upravuje powerProduction.

Farm.cs

Účel:
Jednoduchá trieda pre farmu, ktorú môže rezervovať BioBurner.
Obsahuje len IsReserved(), Reserve(...) a Release().

Vzťahy / Použitie:

- Pre BioBurner, aby mohol navýšiť svoj výkon.

HydroDam.cs (trieda: HydroDam)

Účel:
Spravuje umiestňovanie priehrady. Špeciálny režim v BuildingPlacer.
Dá sa „snapovať“ na body (tag „Hydro“).
Po finalizácii vyvolá TriggerWaterRise() na danom spote, čím sa zvýši hladina vody.

Vzťahy / Použitie:

- Komunikuje s HydroRegistry (vyhľadáva existujúce spoty).
- Používa HydroDamSpot na vyvolanie animácie vody.
- Po finalizácii zničí dočasné indikátory a fixne sa umiestni.

HydroDamSpot.cs

Účel:
Samotný bod, kam sa môže HydroDam snapnúť. Obsahuje odkaz na waterPlane a parametre na „rise animáciu“.
Po spustení TriggerWaterRise() to zapne WaterRaiseController.

HydroRegistry.cs

Účel:
Singleton, ktorý pri spustení vyhľadá všetky objekty s tagom Hydro a uloží si ich do zoznamu.
HydroDam potom z týchto spotov vyberá najbližší.

WaterRaiseController.cs

Účel:
Riadi samotnú animáciu zdvíhania vody (lineárna interpolácia pozície v čase).
StartRising(targetY, duration) spustí korutinu RiseWater(...).

---

4. UI A INTERAKCIA S HRÁČOM

UIManager.cs

Účel:
„Hlavný“ manažér užívateľského rozhrania:

- Zobrazenie / skrytie panelov (ekologická rovnováha, predikovaný výkon a pod.).
- Tooltipy k budovám (využíva BuildingInfoLibrary).

- Animácia slide pre ECO balans, kredity, vyskakovacie texty.

Kľúčové prvky:

- powerStatsBuild, powerSlider, powerFinalPowerText – Pre dočasné zobrazenie predpokladanej produkcie budovy.
- ecoBalanceSlider, ecoBalanceValueText – Zobrazenie eko-balansu.
- creditsText, dynamicCostText – Kredity, cena budovy.
- tooltipPanel (+ polia s textom) – Tooltip pre budovu.
- ShowBuildingTooltip(...) a HideBuildingTooltip(...) – Ovládanie tooltipu.
- ShowEcoLossPopup, ShowCreditGainPopup – Vyskakovacie efekty.

Nové funkcionality (aktualizované o milestone tooltipy a ikony):

- Zobrazenie **ikoniek milestone udalostí** v UI (v hornej časti).
- Tooltip pre udalosť: zobrazuje sa po prechode myšou nad ikonkou.

Metódy (doplnky):

- DisplayMilestoneEvent(MilestoneEventData) → Pridá ikonu udalosti do UI a nastaví tooltip.
- ShowEventTooltip(id), HideEventTooltip() → Práca s tooltipmi milestone udalostí.

Prvky UI (nové):

- eventIconContainer, eventIconPrefab → Miesto pre ikonky.
- eventTooltipPanel, eventTooltipName, eventTooltipClass, eventTooltipDesc → Tooltip panel pre konkrétnu udalosť.

Poznámka:
Funguje podobne ako existujúce tooltipy budov.

BuildPanelController.cs

Účel:
Ovláda ľavý (prípadne iný) panel s možnosťou stavať budovy (animácie, togglovanie, prehľad dostupných budov).

Kľúčové vlastnosti:

- TogglePanel() – Otvára / zatvára panel.
- WaveFadeIn() – Postupne fade-in všetky child UI elementy pre plynulý nábeh.

Vzťahy / Použitie:

- Prepojené s UI tlačidlami (napr. „Build“).
- UIManager môže panel dočasne skryť (napr. pri stavaní budovy).

BuildTabController.cs

Účel:
V rámci build panelu môže byť viacero „tabov“ (napr. „Power Buildings“, „Infrastructure“, atď.). Pomocou SwitchTab(index) vie skryť všetky ostatné a zobraziť len požadovaný obsah.

Kľúčové vlastnosti a metódy:

- currentTabIndex – Aktuálne zvolený tab.
- SwitchTab(int index) – Zapne príslušný tabContent, ostatné vypne.
- FadeInTabContent(GameObject tabContent) – Voliteľná metóda na animovaný prechod.

Vzťahy / Použitie:

- Vnorene používané v BuildPanelController.
- Každý tab obsahuje zoznam tlačidiel budov (prefaby s BuildingButtonHover a pod.).

BuildingButtonHover.cs

Účel:
Umožňuje, aby sa pri prechode myšou nad tlačidlom (UI ikonka budovy) zobrazil tooltip s detailmi.

- Využíva UIManager.Instance.ShowBuildingTooltip(buildingID) a pri opustení HideBuildingTooltip().

Vzťahy / Použitie:

- Implementuje rozhranie IPointerEnterHandler, IPointerExitHandler v Unity.
- Pri pohybe myšou nad tlačidlom volá ShowBuildingTooltip(...) s príslušným buildingID.

SelectionHighlighter.cs

Účel:
Zvýrazňovanie (outline) budov v dosahu, napr. keď ťaháme kábel alebo umiestňujeme budovu.

- HighlightInRange(center, range) – Nájde všetky ConnectableObject v scéne, skontroluje vzdialenosť, a zapne/vypne Outline efekt.

Vzťahy / Použitie:

- ConnectionManager a BuildingPlacer ho využívajú pri ťahaní kábla, resp. pri preview budovy.
- Kedykoľvek potrebujeme zvýrazniť objekty v okolí.

DecalExpander.cs

Účel:
Vizuálny efekt (decal) rozširujúci sa na zemi podľa určitej „pollution“ hodnoty alebo podľa veľkosti budovy.

- Po spustení sa decal lineárne zväčšuje v priebehu expansionDuration.

Vzťahy / Použitie:

- BuildingPlacer ho vytvára, ak má budova ecoCost (napr. znečistenie).
- V spolupráci s EcoBalanceManager možno demonštrovať rozširovanie poškodenia prírody.

RangeIndicator.cs

Účel:
Dynamicky prispôsobuje mesh (zvyčajne kruh) terénu (raycastom) – zobrazuje dosah (napr. veterná turbína, prepojenia).

Vzťahy / Použitie:

- Napr. ConnectableObject.ShowConnectionRangeIndicator() ho aktivuje.
- WindTurbine môže mať vlastný RangeIndicator s tagom „Wind“.

---

5. ZHRNUTIE VZŤAHOV A INTERAKCIÍ

Prehľad

- Budovy (napr. WindTurbine, SolarPanelStatic, BioBurner, HydroDam) obsahujú komponent ConnectableObject, ktorý drží údaje o **produkcii**, **pripojení** a **sieťovom stave**.
- BuildingPlacer rieši umiestňovanie budov („ghost“ režim, kontrola kolízií) a po potvrdení volá:
    - CreditsManager (odpočíta kredity),
    - EcoBalanceManager (odstraňuje vegetáciu, znečistenie).
- ConnectionManager + ConnectionCable riešia pripájanie budov káblami. Pri zmazaní kábla vyvolajú NetworkEvents.TriggerNetworkChanged().

- NetworkManager (PowerNetworkManager) pri každej zmene (cez NetworkEvents) vykoná BFS od hlavného uzla (goal), označí budovy isInNetwork = true, spočíta celkový výkon.
- **UIManager** reaguje na stav hry (kredity, eko balanc, výkon sietí) a zabezpečuje tooltipy, pop-upy, animácie a ostatné časti rozhrania.
- **MilestoneController** sleduje dosiahnutie požadovaného výkonu siete a pri upgrade:
    - odmeňuje hráča kreditmi,
    - **vyberie náhodnú MilestoneEventData** z MilestoneEventLibrary,
    - cez **MilestoneEventManager** aplikuje jej efekty (zvýšenie/zníženie výkonu pre konkrétne budovy).
- **MilestoneEvents** slúži na synchronizáciu pri recalculácii všetkých dotknutých budov. Keď všetky stihnú aktualizovať svoj výkon, spustí sa OnMilestoneRefreshReady, a **MilestoneController** potom vypočíta finálnu odmenu.