

MK1 — architecture et logique game-specific

Positionnement

MK1 est le moteur marketing By Year centré sur création de produit, choix de segment, prix, distribution, communication et lancement sur le marché.

Static data

Les datasets comprennent `scenario`, `segments`, `product_specs`, `shops`, `price_specs`, `medias`, `comms`, `distrib`, `events` et `act_perf_pond`, plus `constants`.

Modèles Unity

- `Product`, `ProductSpecs`, `LaunchedProduct`;
- `Segment`, `Market`;
- `MediaData`, `CommTool`, `DistribData`, `PriceSpec`;
- `LaunchProductSales`, `LaunchProductScores`, `Results`;
- `TeamDecisionsMk1`, `TeamMk1`.

Logique serveur

`GameSpecificUtilsMk1` gère sauvegarde/suppression de produit, lancement, rentabilité, conversion cash↔quantité, sélection de segment, randomisation des décisions bots, événements et création/lancement de produit.

Résultats

```
flowchart LR
P[Produits] --> G[gatherProducts]
SEG[Segments] --> G
G --> AVG[Market averages]
AVG --> PS[Product score]
AVG --> CS[Communication score]
AVG --> DS[Distribution score]
PS --> SALES[Sales]
CS --> SALES
DS --> SALES
SALES --> TEAM[Cash / Results]
```

`GenerateResultsMk1` calcule les moyennes de marché, scores produit/communication/distribution, ventes et résultats annuels.

Bots

`BotsMk1` utilise les heuristiques du moteur MK1 pour générer les décisions d'équipes artificielles en async.

UI

Les écrans spécifiques couvrent portfolio, nouveaux produits, launched products, market, communications, media, shops, calculator, lancement et rapports. Les tutoriels MK1 couvrent notamment création du premier produit et premier lancement.

Ajouter une feature MK1 de A à Z

Entrées de décision principales

`mk1_custom_command.js` route :

- `save_product`;
- `save_launched_product`;
- `delete_product`;
- `delete_launched_product`;
- `delete_calc_product`;
- puis les commandes By Year communes.

Pipeline de résultats

```

flowchart LR
    DEC[Products + launches] --> G[gatherProducts]
    G --> AVG[computeAverages]
    AVG --> PS[getProductScore]
    AVG --> CS[getCommScore]
    AVG --> DS[getDistScore]
    PS --> SCORE[computeScores]
    CS --> SCORE
    DS --> SCORE
    SCORE --> SALES[generateSales]
    SALES --> SAVE[saveResults / finalizeTeamsData]

```

Ajouter un nouveau critère marketing

1. Ajouter la static data (segment/product spec/media/distrib/etc.).
2. Ajouter la propriété dans `Product` ou `LaunchedProduct` selon qu'elle est produit ou lancement.
3. Ajouter le champ UI dans `NewProductControl` / `LaunchProductControl` ou un nouveau contrôle.
4. Transmettre la valeur via `save_product` ou `save_launched_product`.
5. Ajouter le coût dans `getProductCost()` / `getLaunchProductCost()` si applicable.
6. Ajouter la valeur au score pertinent : produit, comm ou distribution.
7. Vérifier averages et normalisation si le score est relatif aux concurrents.
8. Vérifier `generateSales()`.
9. Ajouter le report Unity et éventuellement ActPerf/ranking.
10. Adapter `BotsMk1.takeDecisions()` si le critère est requis.

Calculateur vs décision réelle

MK1 contient des produits de calculateur en plus des produits réellement sauvegardés/lancés. Une nouvelle propriété doit être cohérente dans les deux chemins si le calculateur est censé la simuler.

Auditer `saveProduct`, `saveLaunchedProduct` et `deleteCalcProduct`.

Debug ventes MK1

1. produit/lancement persisted;
2. segment cible;
3. averages marché;
4. product score;
5. comm score;
6. distribution score;
7. sales generated;
8. cash/result saved;
9. report Unity.

Revision #2

Created 2026-09-18 19:21:52 UTC by Mathis

Updated 2026-09-18 20:19:25 UTC by Mathis