Minegicka Legacy → Forge 1.20.1

Récapitulatif des changements (mini-rapports Codex)

1) Arborescence & ressources standardisées

- Fusion de l'arbre legacy en **une seule source** : `src/main/java/...` ; suppression des doublons `java/` et `williameze/java/...`.
- Ressources replacées sous `src/main/resources/...` (suppression des anciens emplacements).
- Objectif atteint : Gradle ne voit **qu'un seul** arbre standard.

2) Toolchain & build

- **Gradle → JDK 17** par `org.gradle.java.home` ; compilation passe nettement plus loin.
- `./gradlew compileJava` **OK**. Seul avertissement résiduel : *"plug-in not found: ErrorProne"* pendant le bootstrap Forge (non bloquant à ce stade).

3) Squelette Forge 1.20 moderne

- Nouveau **entrypoint** mod avec **IEventBus**, wiring config, **DeferredRegister**, et **DistExecutor** (proxy commun/client) pour remplacer CommonProxy/ClientProxy.
- Stubs d'événements Forge (ticks, hooks) prêts à recevoir la logique gameplay.

4) Couches de registries

- Découpage GameRegistry → **DeferredRegister** : items, blocks, mob effects, entity types.
- **Creative Tab** alimenté par les registries. Placeholders prêts à être remplacés par des ports réels.

5) Lancement dev client

- Correctif **JPMS auto-module inference** dans `build.gradle` pour éviter les collisions d'exports.
- `runClient` démarre proprement ; avertissements attendus sur **JSON modèles manquants** tant que la DataGen n'émet pas d'assets.

6) Capability Joueur & persistance

- Ajout de **PlayerManaCapability** + **PlayerManaData** (récupération/consommation, déblocages, dirty tracking).
- **SavedData** serveur et **events** (login/logout/dimension/tick) pour persister l'état.
- **Networking**: SimpleChannel + **PlayerManaSyncPacket** (sync S→C lorsque dirty).

7) Magicks, items et overlay

- **MagickRegistry** câblé (catalogue, id/combo/cout de base) pour préparer le gameplay.
- Items portés : **StaffItem**, **MagickTabletItem** (stats, unlock/cast), **ItemManaHelper**.
- **SpellCasting** connecté à la capability (contrôles mana + déclencheurs d'effets légers).
- **Mana Overlay** client simple ; **DataGen** amorcée (modèles, lang) pour réduire les warnings d'assets.

État actuel (snapshot)

- `compileJava` ✅
- `runClient` ✅(warnings d'assets tant que datagen/assets incomplets)
- Datagen : squelette présent, **à compléter**
- Avertissement outillage : **ErrorProne** non résolu côté bootstrap Forge (non bloquant)
- Legacy `PlayersData`/maps globales : **capability en place** → nettoyage final à planifier après câblage complet des appels

---

Roadmap priorisée (Étapes 4 → 6)

4.1 — Datagen & assets minimum

But : éliminer les warnings de modèles et stabiliser l'itération. - Modèles JSON items/blocks, lang, loot de base pour les entrées déjà enregistrées. - Brancher les providers (item models, block states, loot, recipes, lang) et valider `runData`. **DoD** : `runData` génère sans erreur ; `runClient` ne logge plus d'assets manquants pour le contenu enregistré. **Cmd** : `./gradlew runData && ./gradlew runClient`

4.2 — Intégration capability 100% (nettoyage PlayersData)

But : retirer les structures globales legacy. - Remplacer tous les usages `PlayersData/PlayerData` par la capability et helpers. - Retirer les anciennes sauvegardes NBT ad-hoc. **DoD** : plus aucune référence aux maps globales ; mort/déconnexion/dimension → état intact.

4.3 — Réseau : port complet des packets

But : rétablir la synchro et les actions gameplay. - Migrer les ~packets restants vers **SimpleChannel** (encode/decode/handler explicites, versioning). - Ajout d'un **handshake** versionné et d'un mini schéma d'ACK/logging. **DoD** : tout flux gameplay (cast, unlock, HUD) transite via SimpleChannel sans erreurs.

4.4 — Vertical slice gameplay (spell + entité + rendu)

But : prouver le pipeline end-to-end. - Porter **1 sort** (ex. *beam*) + **1 entité magique** (ex. `EntityBeam`) : `EntityType`, attributes, tick. - Enregistrement **renderer** via `EntityRenderersEvent.RegisterRenderers` ; suppression des anciens `RenderingRegistry` & TESR. - Lier **SpellCasting** → entité & effets, consommation **mana** → overlay. **DoD** : en jeu, on cast le sort, on voit l'effet, la mana baisse et se resynchronise.

4.5 — Staffs réels & équilibrage

But : sortir du placeholder. - Sous-classes de Staffs (stats, NBT, capabilities si besoin). - Récupération mana via stats tenues (déjà amorcé) + recettes. **DoD** : 2–3 staffs complets avec recettes & modèles.

4.6 — Datagen complète (recipes/tags/loot/lang)

But : zéro warning au chargement, contenu reproductible. - Couvrir l'intégralité des items/blocks/entités migrés ; tags (ex. `forge:rods/wooden`, etc.). - Ajouter une vérif CI locale : échec si assets manquants. **DoD** : `runClient` **zéro** warning d'assets ; `runData` idempotent.

4.7 — HUD & UX

But : feedback clair pour testeurs. - Keybinds (cast, quick-select), HUD mana plus riche, sons/particles de base. - Option config (client) pour activer/désactiver l'overlay. **DoD** : actions testables sans console, overlay lisible.

4.8 — Qualité & outillage

But : pérenniser la migration. - Décider : **vendoriser ErrorProne** (classpath) **ou** le désactiver sur la phase FG incriminée. - **Spotless** + `.editorconfig` appliqués ; **Semgrep** règles "legacy→moderne" ; premières recettes **OpenRewrite** (ex. `GameRegistry.* → DeferredRegister`). - **Tests unitaires** : sérialisation capability, logique mana. **DoD** : build reproductible, formatage stable, tests verts.

4.9 — Worldgen / Blocks spé / Block Entities (après slice validé)

But : étendre progressivement. - Migration tuiles/renders via `BlockEntityRenderers` ; worldgen aux APIs 1.20 (placed/configured features). **DoD** : au moins un bloc "complexe" migré avec rendu et loot cohérents.

Check-list exécutable

- [ ] `./gradlew runData` sans erreur
- [ ] `./gradlew runClient` sans warnings d'assets
- [ ] Suppression totale de `PlayersData` & co.
- [ ] Tous les packets migrés → `SimpleChannel`
- [ ] 1 sort + 1 entité magique + renderer → **OK en jeu**
- [ ] 2–3 staffs complets (NBT, recettes, modèles)
- [ ] Spotless/EditorConfig/Semgrep en place ; recettes OpenRewrite initiales
- [ ] Tests unitaires capability verts

Bonnes pratiques & jalons de commit

- Commits **petits et atomiques** : *"registry: add StaffItem + datagen"*, *"net: port PlayerManaSyncPacket"*, etc.
- Branche par slice (`feat/slice-spell-beam`, `feat/datagen-baseline`).
- À chaque jalon : *compile* → *runData* → *runClient* ; si OK, tag interne (`v0.1-slice1`).

Commandes utiles (rappel)

```
# Générer les run configs IntelliJ si besoin
./gradlew genIntellijRuns

# Réécriture / formatage (lorsqu'activés)
./gradlew rewriteRun spotlessApply

# Build rapide
./gradlew clean compileJava

# Données & client
a) ./gradlew runData
b) ./gradlew runClient
```

Notes de suivi

- L'avertissement ErrorProne est **toléré** pour l'instant (Bootstrap Forge). Décision à prendre en 4.8.
- Dès que la capability est branchée partout, **supprimer** le code legacy équivalent pour éviter la dérive.
- Penser à coller une **texture placeholder** générique (16×16) pour tous les items en WIP afin de limiter le bruit de logs.