Rapport sur l'utilisation du langage Faust (Functional Audio Stream)

Date : 17 mai 2025

---

1. Introduction

Faust est un langage de programmation fonctionnel dédié au traitement du signal audio en temps réel. Vingt-trois ans après sa première version publique (2002), Faust est devenu un pilier de la recherche DSP et un outil de production utilisé aussi bien dans des pédales de guitare que dans des applications Web. Le présent rapport synthétise les principaux indicateurs d'adoption et de performance disponibles publiquement ; il peut servir de base pour le suivi annuel du projet ou pour alimenter un dossier d'investissement.

---

2. Indicateurs d'adoption « macro »

Source	Métrique	Valeur (17 mai 2025)	Commentaire
GitHub	⋆ Stars	**2 700+**	Croissance régulière (~ +6 %/an depuis 2022) (github.com)
	Forks	**348**	Montre l'activité des forks (ex. `faustwasm`, `faust-rs`) (github.com)
	Issues ouvertes	**≈ 200**	Taux de fermeture : 75 % dans l'année écoulée (github.com)
Homebrew	Installations 30 jours	**156**	Représente macOS + Linux ; légère hausse après la sortie 2.79.3 (formulae.brew.sh)
	Installations 90 jours	**652**	58 % d'installations « on-request » (usage explicite) (formulae.brew.sh)
	Installations 365 jours	**2 568**	+14 % vs. période précédente (formulae.brew.sh)

Source	Métrique	Valeur (17 mai 2025)	Commentaire
Projets recensés	Entrées « Powered By Faust »	**> 200**	Plugins, matériels, applis mobiles, sites Web, etc. (faust.grame.fr)
Publications académiques	Articles listés	**≈ 160**	Bibliographie officielle maintenue par GRAME (faust.grame.fr)

> **À retenir :** les KPI externes confirment une base d'utilisateurs modeste mais engagée ; la progression la plus nette vient des projets embarqués (Daisy, Bela) et des plugins Rust/VST3.

---

3. Zones de croissance identifiées

1. **Écosystème Web** : le package `@grame/faustwasm` est mis à jour toutes les 6 semaines ; les téléchargements npm mensuels sont estimés à plusieurs milliers (API npm injoignable pour un chiffre exact au moment de la rédaction).
2. **Intégrations industrielles** : adoption croissante dans les frameworks JUCE / HISE ; plusieurs produits commerciaux (Chaos Audio Stratus, Expressive E Noisy 2) citent Faust comme moteur DSP principal.
3. **Éducation & recherche** : l'International Faust Conference (IFC) rassemble ~120 participants/édition ; la liste des publications croît d'une dizaine par an.

---

4. Indicateurs « micro » : performance des DSP

4.1 Outils de benchmarking

Outil	Objectif	Sorties principales
`faust2bench` / `faustbench-llvm`	Mesure du CPU % et du débit mémoire	CSV : *avg CPU *, *throughput*

Outil	Objectif	Sorties principales
Option `-time` du compilateur	Chronométrage détaillé de chaque phase	Log console
`interp-tracer`	Profil numérique (NaN, overflows…)	Rapport texte

4.2 Exemple de workflow de suivi

1. **Intégration CI** : exécuter `faustbench-llvm` sur chaque commit majeur ; exporter le CSV.
2. **Stockage** : envoyer le CSV dans Prometheus ou InfluxDB.
3. **Visualisation** : dashboard Grafana avec alertes (> x % de régression CPU).

---

5. Recommandations

- **Mise en place d'un tableau de bord public** combinant : GitHub API (stars/forks), Homebrew Analytics, npm downloads, et flux RSS des publications. Un refresh hebdomadaire suffit.
- **Surveillance des performances** : standardiser l'usage de `faustbench-llvm` dans les forks et dans les scripts `faust2*`; publier les scores dans les releases.
- **Communauté** : encourager les mainteneurs de projets listés dans « Powered By Faust » à ajouter des *badges* GitHub (Built with Faust) pour accroître la visibilité.

---

6. Conclusion

Avec une communauté GitHub active, une adoption en hausse sur Homebrew et plus de deux cents projets référencés, **Faust** poursuit sa trajectoire d'outil clé pour la création audio temps-réel. La prochaine étape logique est la consolidation des métriques (tableau de bord unifié) afin de piloter plus finement les priorités du roadmap – en particulier le compilateur WebAssembly et les cibles embarquées.
