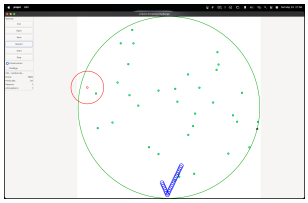
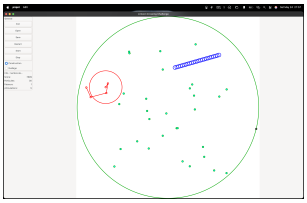
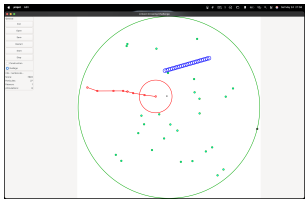
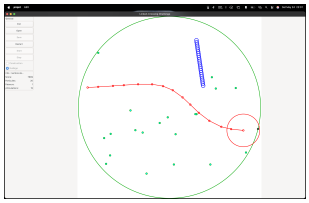
Évolution de l'Organisation du Code et Exécution

La structure modulaire **Contrôle-Modèle-Vue adaptée** — modules `jeu`, `mobile`, `chaine`, `tools`, `gui`, `graphic` — est restée inchangée entre les rendus 2 et 3, tout comme la hiérarchie `Mobile → Particule/Faiseur`. Le travail s'est concentré sur la **finalisation de la logique de jeu** : interactions avancées de la chaîne, détection de victoire/défaite et sérialisation fiable.

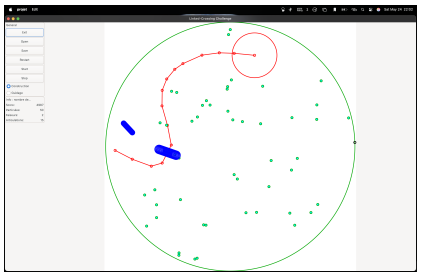
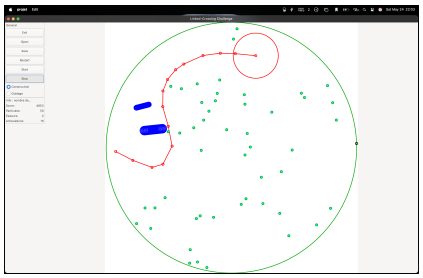
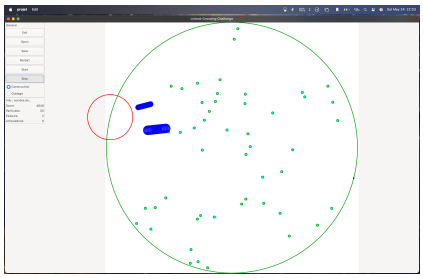
Tests principaux

`t22.txt` — *Construction & Guidage*

1. **Début** : 1 articulation capturée ; cercle de capture rouge et but noir visibles ; score -1.	2. **Chaîne à 5 articulations** : cercle recentré sur le nouvel effecteur.	3. **Chaîne à 8 articulations + FABRIK** : but intermédiaire défini par clic droit, forme ajustée ; retour en mode CONSTRUCTION.	4. **Chaîne à 15 articulations** : alternance guidage/construction pour éviter obstacles et capturer une particule précise.

`t35.txt` — *Destruction par Faiseur*

1. **État initial** : 15 articulations, 2 faiseurs, score 4987.	2. **Collision imminente** : score 4853 (écart d'une étape vs. consigne).	3. **Destruction** : articulation incluse dans un faiseur → chaîne remise à 0, score 4849, zone de capture réinitialisée. :

Cette campagne de tests valide la robustesse du moteur physique, la cohérence score-événements et l'ergonomie des modes CONSTRUCTION/GUIDAGE.

Méthodologie de Travail, Contributions et Conclusion

Activité individuelle et usage de l'IA

ChatGPT et *Gemini 2.5* ont servi d'assistants de débogage (recherche d'incohérences, suggestions d'optimisation) ; leur apport direct en code exécutable est resté limité.

Jean Aboutboul (375057)

- *Individuel*
  - Cœur de `jeu.cc` : états, boucle `updateJeu`, parser fichiers, sauvegarde/chargement.
  - Hiérarchie `mobile.*` : logique d'update/dessin/sérialisation, réflexions aux bords.
  - Simulation particules/faiseurs, détection de mouvement légal (`canFaiseurMove`).
  - Géométrie fine des queues de faiseurs.
- *En groupe*
  - Conception API `tools.cc`, intégration modules, définition constantes.

Perla Brodowicz (390880)

- *Individuel*
  - Modules `gui.*`, `graphic.*` : widgets GTKmm, signaux, timer, dialogues fichiers.
  - `on_draw` : échelle homogène, inversion Y, affichage infos latérales.
  - Adaptation `chaine.*` : construction, FABRIK via `Jeu::guidanceStep`, dessin complet.
- *En groupe*
  - Interface GUI↔Modèle, tests d'intégration, séquence de rendu.

Travail collaboratif

- **Gestion de version :** Git/GitLab, branches courtes, revues croisées quotidiennes (≈ 70 % travail conjoint, 30 % tâches isolées).
- **Environnement :** VS Code + débogueur intégré.
- **Tests** : unitaires (`tools`, `mobile`) dans `test.cc`, puis intégration sur jeux fournis et scénarios limites.

Bugs notables et résolutions

- Invalidation d'itérateurs lors de suppressions dans `std::vector` $\rightarrow$ parcours inverse ou indices tampon.
- Post-conditions mal précisées entre modules $\rightarrow$ doc renforcée et assertions.
- Réflexion précise de la queue des faiseurs après multiples rebonds $\rightarrow$ traçage papier + pas-à-pas GDB.

Bilan

L'approche par rendus progressifs a encouragé les bonnes pratiques de développement. Pour améliorer encore le projet, des scripts de validation automatique intermédiaires pour les calculs géométriques auraient réduit le temps de débogage initial.

Conclusion :projet formateur et abouti, combinant algorithmique, conception logicielle et interface utilisateur dans un cadre motivant.