Fiche descriptive de projet Hub

Nom du projet : Spritly

Contexte et but du projet

Le projet Spritly naît du besoin croissant d'outils de création graphique accessibles et performants pour les développeurs de jeux vidéo, designers et artistes numériques. Les éditeurs de sprite sheets existants sont souvent complexes, nécessitent une installation lourde ou manquent de fonctionnalités modernes pour l'animation frame par frame.

Spritly vise à créer un éditeur de sprite sheets moderne, léger et accessible directement depuis un navigateur web. L'application permettra aux utilisateurs de créer, éditer et animer des sprites frame par frame de manière intuitive, avec une interface responsive et des outils de dessin performants basés sur l'API Canvas. Bien que l'outil soit principalement conçu pour créer des sprite sheets animées, il permet également la création de sprites simples.

L'usage final consiste en une application web hébergée sur GitHub Pages, accessible gratuitement via un simple lien, permettant à quiconque de créer des animations de sprites sans installation préalable. L'outil sera particulièrement adapté aux développeurs indépendants, aux étudiants d’Epitech et aux pixel artists souhaitant créer des animations pour leurs jeux.

Porteur(s) du projet

Équipe de développement :

- **Alexandre Bacha** : Développeur, responsable de l'architecture frontend React et de l'intégration Canvas API
- **Aurélien Lamberger** : Développeur, responsable du développement des outils de dessin et de l'optimisation des performances

Partenaires externes : Aucun partenaire externe n'est impliqué pour le moment. Le projet est réalisé en totale autonomie.

Objectif fonctionnel

User Stories principales

En tant qu'utilisateur, je souhaite :

1. **Créer un nouveau sprite** : Je peux initialiser un canvas vierge avec des dimensions personnalisables (16x16, 32x32, 64x64, ou dimensions personnalisées)
2. **Dessiner avec des outils basiques** : Je peux utiliser un crayon, une gomme, un pot de peinture (remplissage), et un sélecteur de couleur pour créer mon sprite pixel par pixel
3. **Gérer les calques** : Je peux créer, supprimer, réorganiser et basculer la visibilité de plusieurs calques pour organiser mon travail
4. **Utiliser une palette de couleurs** : Je peux sélectionner des couleurs via un sélecteur de couleur, sauvegarder mes couleurs favorites et gérer une palette personnalisée
5. **Prévisualiser mon sprite** : Je peux voir un aperçu en temps réel de mon sprite à différentes échelles pour vérifier le rendu final
6. **Annuler/Refaire mes actions** : Je peux annuler (Ctrl+Z) ou refaire (Ctrl+Y) mes modifications pour corriger des erreurs
7. **Exporter mon travail** : Je peux exporter mon sprite en PNG avec transparence, dans différentes résolutions

8. **Sauvegarder et charger des projets** : Je peux sauvegarder mon projet localement (JSON) et le recharger ultérieurement pour continuer mon travail
9. **Utiliser des raccourcis clavier** : Je peux accéder rapidement aux outils via des raccourcis clavier pour améliorer ma productivité
10. **Travailler sur mobile** : L'interface s'adapte aux écrans tactiles pour permettre une utilisation sur tablette

Environnement technique / technologique

Stack technique :

- **Langage** : TypeScript (pour le typage statique et la maintenabilité)
- **Framework frontend** : React 18+ (avec hooks pour la gestion d'état)
- **API graphique** : Canvas API (pour le rendu et la manipulation des pixels)
- **Build tool** : Vite (pour des temps de compilation rapides)
- **Gestionnaire de versions** : Git + GitHub
- **Hébergement** : GitHub Pages (déploiement statique)

Bibliothèques complémentaires envisagées :

- Tailwind CSS ou styled-components (pour le styling)
- Zustand ou Context API (pour la gestion d'état globale)
- file-saver (pour l'export de fichiers)

Environnement de développement :

- Node.js (v18+)
- npm ou yarn (gestion des dépendances)
- VS Code

Contraintes techniques :

- Application 100% client-side (pas de backend)
- Compatible avec les navigateurs modernes (Chrome, Firefox, Safari, Edge)
- Performance optimisée pour des canvas jusqu'à 512x512 pixels

- Responsive design pour écrans desktop et tablettes

Description du livrable

Éléments du livrable :

1. **Application web déployée**
    a. URL publique sur GitHub Pages
    b. Application monopage (SPA) entièrement fonctionnelle
    c. Documentation utilisateur intégrée (page d'aide ou tooltips)
2. **Code source**
    a. Repository GitHub public avec historique Git propre
    b. Architecture modulaire avec composants React réutilisables
    c. Code TypeScript typé et documenté
    d. Configuration Vite pour le build et le déploiement
3. **Système de build**
    a. Scripts npm pour le développement, build et déploiement
    b. Pipeline GitHub Actions pour le déploiement automatique
4. **Documentation technique**
    a. README.md complet avec instructions d'installation et utilisation
    b. Documentation des composants principaux
    c. Guide de contribution pour d'éventuels contributeurs
5. **Assets et ressources**
    a. Icônes d'outils (crayon, gomme, pipette, etc.)
    b. Sprites de démonstration
    c. Favicon et métadonnées pour le partage social

Niveau de finition attendu :

- Application déployée et accessible 24/7
- Interface utilisateur complète et fonctionnelle
- Tests manuels validés sur différents navigateurs
- Documentation utilisateur et technique complète
- Code source propre

Organisation et temporalité

Phase 1 : Configuration et fondations (20h)

- Semaines 1-2
- Setup du projet (Vite + React + TypeScript)
- Configuration GitHub et GitHub Pages
- Architecture des composants de base
- Implémentation du Canvas de base

Phase 2 : Outils de dessin fondamentaux (30h)

- Semaines 3-5
- Implémentation du crayon et de la gomme
- Système de gestion des couleurs
- Sélecteur de couleurs
- Zoom et navigation dans le canvas

Phase 3 : Fonctionnalités avancées (35h)

- Semaines 6-8
- Système de calques complet
- Outil de remplissage (bucket fill)
- Système undo/redo
- Outils supplémentaires (ligne, rectangle, cercle)

Phase 4 : Import/Export et sauvegarde (20h)

- Semaines 9-10
- Export PNG avec options
- Sauvegarde/chargement de projets (JSON)
- Import d'images existantes

Phase 5 : Polish et optimisation (21h)

- Semaines 11-12
- Amélioration de l'UI/UX
- Optimisation des performances Canvas
- Raccourcis clavier

- Responsive design

Dépendances critiques :

- Phase 2 dépend de la Phase 1 (fondations Canvas)
- Phase 3 dépend de la Phase 2 (outils de base)
- Phase 4 peut être développée en parallèle de la Phase 3

Volume horaire total : 125 heures

- Alexandre Bacha : 63 heures
- Aurélien Lamberger : 63 heures

Jalons importants :

- Fin Phase 1 : Premier déploiement sur GitHub Pages
- Fin Phase 2 : MVP fonctionnel avec outils de base
- Fin Phase 3 : Application complète avec features avancées
- Fin Phase 5 : Livraison finale et présentation du projet