Projet: Pixel Paint - Outil de création de sprites 32x32 pour NES/GB

URL: https://github.com/Kully/pixel-paint

URL de démo: https://kully.github.io/pixel-paint/

Membres du groupe:

Nom	N° etudiant	GitHub
Viet Nguyen	20006303	Viet281101
Raphaël MOSCATELLI	21006543	Raphaelmos

Description:

- Application web créée avec JavaScript permettant de dessiner des sprites de 32x32 pixels avec les palettes de couleurs des consoles NES et Gameboy.
- L'application permet de dessiner des sprites de 32x32 pixels sur un canvas.
- L'utilisateur peut choisir parmi les palettes de couleurs de la NES (62 couleurs et 4 niveaux de gris).
- Les outils disponibles sont les pinceaux de diverses tailles (1x1, 2x2 etc), effacement partiel/total et export de l'image.

Technologies utilisées:

- Langages: HTML, CSS, JavaScript
- Bibliothèques: Pas de bibliothèque externe
- Pas d'API externe ou d'infrastructure nécessaire

Le projet est en vanilla JavaScript, sans bibliothèque externe.

Outils de développement:

- Gestionnaire de versions: Git et hébergé sur GitHub
- Environnement de développement: éditeur de code (non spécifié)
- Déploiement: GitHub Pages
- Debogage: Console du navigateur et outils de développement
- Rendu: Navigateur web

Organisation du code:

Fichiers JavaScript dans le dossier `js/`

Assets dans le dossier `img/`

Demo dans le dossier `gif/`

- `index.html`:
  - Contient la structure HTML de base et importe les styles et scripts.
- `style.css`:
  - Définit le style et mise en page de l'application (toolbox, canvas).
- `script.js`:
  - Initialise le canvas et les variables globales (couleurs, outils).
  - Gère les événements clic/déplacement sur le canvas pour le dessin.
  - Permet de changer d'outil et d'effacer.
  - Exporte l'image en PNG.
- `utils.js`:
  - Contient des fonctions utilitaires pour le dessin.
- `tools.js`:
  - Définit les classes des différents outils (pinceaux).
  - Gère leur taille, couleur et fonctionnalités de dessin.
- `palette.js`:
  - Définit les tableaux de couleurs NES et GB.
  - Permet de sélectionner la palette utilisée.
- `historyState.js`:
  - Permet de gérer l'historique des états du canvas pour les fonctions undo/redo.

Le canvas se situe dans le body, entouré de la toolbox avec les boutons de couleurs et outils. Le code est clairement modularisé entre gestion des couleurs, outils et logique métier.

Conventions de codage:

- Indentation à tabulations
- Nommage explicite et en anglais
- Commentaires décrivant les fonctions
- Chaînes de caractères entre quotes simples
- Respect des standards W3C pour le HTML/CSS
- Utilisation de `let` et `const` pour les variables
- Les fonctions sont déclarées avec `function` et le nom de la fonction est en `snake_case` avec le premier mots en majuscule (par exemple: Add_EventHandlers_To_Canvas_Cells())
- Les constantes sont en majuscules (par exemple: `const MAX_UNDO = 10;`)

- Les classes sont en `PascalCase` (par exemple: `class HistoryState`)
- Les variables sont en `camelCase` (par exemple: `let currentColor`)

Fonctionnalités:

- Effacement partiel/total
- Export de l'image en PNG
- Palette de couleurs NES (62 couleurs + 4 niveaux de gris)
- Palette de couleurs Game boy (4 couleurs)
- Undo/Redo
- Outil de remplissage
- Outil de sélection
- Outil de pipette
- Outil de copie/déplacement
- Outil de suppression de la sélection
- Outil de remplacement de couleur
- Grille de 32x32 pixels
- Curseurs et icônes originaux 32x32
- Raccourcis clavier pour les outils
- Raccourcis clavier pour les actions (undo, redo, etc)

Contributing

Issues:

- Line is drawn choppy on fast mouse movements #8
- Export to hex colors #10

Contributeurs actuels:

- Kully
- Lucas Ilari
- R-Lovelett
- aleksi100