Rapport de Développement Logiciels Libres

Informations générales

Nom du projet	Compte GitHub	Propriétaire du projet	Pull Request	Resolve l'issue	Objectif
Pixel Paint	Viet281101, Raphaelmos	Kully	Fix the line draw choppy on fast mouse movements issue #20	Line is drawn choppy on fast mouse movements #8	Améliorer la fluidité du tracé des lignes avec une souris en déplacement rapide.

Description détaillée

Problème

- **Symptôme**: Les lignes tracées sont saccadées lorsque la souris se déplace rapidement.
- **Cause**: L'algorithme actuel de dessin n'est pas optimisé pour les mouvements rapides de la souris.

Solution

1. **Algorithme de Bresenham**:
    - **Raison du choix**: L'algorithme de Bresenham est particulièrement adapté pour le tracé de lignes droites dans les graphiques raster. Contrairement à la régression linéaire ou d'autres méthodes, il est rapide, efficace et n'implique pas de calculs en virgule flottante, ce qui le rend parfait pour une application en temps réel dans un contexte de dessin pixelisé.
    - **Mise en œuvre**: L'algorithme de Bresenham a été implémenté pour améliorer la fluidité du tracé en gérant les déplacements rapides de la souris.
    - Cet algorithme permet de tracer des lignes plus précises, réduisant les saccades lors des mouvements rapides de la souris.

```
/**
 * Draws a line using the Bresenham's line algorithm. The line is drawn
from (startX, startY) to (endX, endY)
 *
 * @param {number} startX - The x-coordinate of the starting point.
 * @param {number} startY - The y-coordinate of the starting point.
 * @param {number} endX - The x-coordinate of the ending point.
 * @param {number} endY - The y-coordinate of the ending point.
 * @param {function} callback - The callback function to be called for each
cell on the line.
 * @return {void} This function does not return a value.
 */
function Bresenham_Line_Algorithm(startX, startY, endX, endY, callback)
{
```

```
    if (typeof callback !== 'function') { console.error("Invalid callback
function"); return; }
    let deltaX = Math.abs(endX - startX);
    let deltaY = Math.abs(endY - startY);
    let stepX = (startX < endX) ? 1 : -1;
    let stepY = (startY < endY) ? 1 : -1;
    let error = deltaX - deltaY;

    while (true)
    {
        let cellId =
Pad_Start_Int(Get_CellInt_From_CellXY(Math.round(startX),
Math.round(startY)));
        let currentCell = document.getElementById(cellId);

        if (currentCell) {
            callback(currentCell);
        }
        if (Math.round(startX) === Math.round(endX) &&
            Math.round(startY) === Math.round(endY)) break;

        let error2 = 2 * error;
        if (error2 > -deltaY) {
            error -= deltaY;
            startX += stepX;
        }
        if (error2 < deltaX) {
            error += deltaX;
            startY += stepY;
        }
    }
};
```

2. **Réorganisation du code**:
    - Séparation des fonctions de gestion des événements (EventHandlers) dans un fichier distinct.
    - Optimisation de la structure du code pour faciliter la maintenance et l'extension futures.

Processus de réalisation

1. **Analyse du problème**:
    - Étude du phénomène de saccades lors du tracé avec des mouvements rapides de la souris.
    - Recherche et compréhension de l'algorithme de Bresenham pour son application au projet.
2. **Amélioration du code**:
    - Réécriture du code de tracé des lignes en utilisant l'algorithme de Bresenham.
    - Réorganisation des fonctions de gestion des événements dans un fichier dédié pour améliorer la clarté et la gestion du code.
3. **Tests et évaluation**:

- Tests des outils de crayon et de gomme pour assurer leur bon fonctionnement avec les modifications.
- Évaluation des performances et de la fluidité du tracé après l'application du nouvel algorithme.

Problèmes supplémentaires résolus

1. Tracé interrompu à la sortie rapide de la zone de dessin:
   - Utilisation de l'algorithme de Bresenham pour relier le point final du tracé à la bordure du canvas.

```
canvasDiv.addEventListener("mouseleave", function (e) {
    if (STATE["brushDown"]) {
        const canvasRect = canvasDiv.getBoundingClientRect();
        const x = Math.max(0, Math.min(e.clientX - canvasRect.left,
canvasRect.width - 1));
        const y = Math.max(0, Math.min(e.clientY - canvasRect.top,
canvasRect.height - 1));
        const targetCell = document.elementFromPoint(x + canvasRect.left, y
+ canvasRect.top);

        if (targetCell && targetCell.classList.contains('canvasCell')) {
            const targetX = targetCell.offsetLeft / CELL_WIDTH_PX;
            const targetY = targetCell.offsetTop / CELL_WIDTH_PX;
            // console.log(`Previous Cursor: (${previousCursorX},
${previousCursorY}) Exit Target: (${targetX}, ${targetY})`);
            if (previousCursorX !== null && previousCursorY !== null) {
                Bresenham_Line_Algorithm(previousCursorX, previousCursorY,
targetX, targetY, Get_Tool_Action_Callback());
            }
            previousCursorX = previousCursorY = null;
        }
        document.addEventListener("mousemove", Track_Mouse_Outside);
    }
    isDrawingOutside = true;
});
```

2. Tracé interrompu à la réentrée rapide dans la zone de dessin:
   - Suivi des événements de la souris et calcul du vecteur de déplacement pour déterminer le point d'entrée (`entryPoint`) et le relier au point cible (`targetPoint`) avec Bresenham.

```
canvasDiv.addEventListener("mouseenter", function (e) {
    if (STATE["brushDown"] && isDrawingOutside) {
        isDrawingOutside = false;
        const targetCell = document.elementFromPoint(e.clientX, e.clientY);

        if (targetCell && targetCell.classList.contains('canvasCell')) {
            const targetX = targetCell.offsetLeft / CELL_WIDTH_PX;
            const targetY = targetCell.offsetTop / CELL_WIDTH_PX;
```

```
            let entryPointX, entryPointY;
            const deltaX = targetX - lastOutsideX;
            const deltaY = targetY - lastOutsideY;

            if (Math.abs(deltaX) > Math.abs(deltaY)) {
                if (deltaX > 0) { entryPointX = 0; }
                else { entryPointX = CELLS_PER_ROW - 1; }
                entryPointY = Math.floor(lastOutsideY);
            } else {
                if (deltaY > 0) { entryPointY = 0; }
                else { entryPointY = CELLS_PER_ROW - 1; }
                entryPointX = Math.floor(lastOutsideX);
            }
            entryPointX = Math.max(0, Math.min(entryPointX, CELLS_PER_ROW -
1));
            entryPointY = Math.max(0, Math.min(entryPointY, CELLS_PER_ROW -
1));

            if (entryPointX === 0 || entryPointX === CELLS_PER_ROW - 1) {
                entryPointY = Math.floor(lastOutsideY);
            } else if (entryPointY === 0 || entryPointY === CELLS_PER_ROW -
1) {
                entryPointX = Math.floor(lastOutsideX);
            }
            // console.log(`Entry Point: (${entryPointX}, ${entryPointY})
Target: (${targetX}, ${targetY})`);
            Bresenham_Line_Algorithm(entryPointX, entryPointY,targetX,
targetY, Get_Tool_Action_Callback());

            previousCursorX = targetX;
            previousCursorY = targetY;
            document.removeEventListener("mousemove", Track_Mouse_Outside);
        }
    }
});
```

3. Problème de tracé aux coins du canvas:
    - Ajustement des calculs de vecteur pour s'assurer que les points aux coins (0 ou 31) soient corrects, évitant ainsi les erreurs de tracé.

```
if (Math.abs(deltaX) > Math.abs(deltaY)) {
    if (deltaX > 0) { entryPointX = 0; }
    else { entryPointX = CELLS_PER_ROW - 1; }
    entryPointY = Math.floor(lastOutsideY);
} else {
    if (deltaY > 0) { entryPointY = 0; }
    else { entryPointY = CELLS_PER_ROW - 1; }
    entryPointX = Math.floor(lastOutsideX);
}
```

```
entryPointX = Math.max(0, Math.min(entryPointX, CELLS_PER_ROW - 1));
entryPointY = Math.max(0, Math.min(entryPointY, CELLS_PER_ROW - 1));
```

4. Corrections de syntaxe et optimisation du code:
    - Ajustements selon les commentaires de `Kully`: conversion des tabulations en espaces, renommage des fonctions, ajout de `JSDoc` pour l'algorithme de Bresenham.
    - Nettoyage du code, suppression des commentaires inutiles, et correction des erreurs dues aux nouvelles implémentations dans eventHandlers.js.

Interaction et feedback

- **Feedback de `Kully`**:
    - Remarques sur l'amélioration de la fluidité du tracé.
    - Suggestions d'ajustements mineurs pour optimiser davantage le code.
- **Réponses et actions**:
    - Remerciements pour les retours et conseils.
    - Ajustements selon les suggestions de `Kully` et mise à jour de la Pull Request.

Résultats

- **Tracé plus fluide**: L'algorithme de Bresenham améliore considérablement l'expérience de dessin lors des mouvements rapides de la souris.
- **Code plus propre**: La réorganisation des fonctions de gestion des événements rend le code plus facile à gérer et à étendre.

Conclusion

Cette Pull Request a résolu efficacement le problème de saccades lors du tracé rapide, tout en améliorant la structure du code du projet Pixel Paint. Les modifications apportées améliorent non seulement l'expérience utilisateur, mais facilitent également la maintenance et le développement futur du projet. Le Pull Request a été fusionné et l'issue #8 a été fermée.