Imagerie Numérique : TD 3 (sur machine)

Opacité et transparence, manipulation d'images, Javascript : créer et effacer des balises

`http://www.labri.fr/perso/preuter/in2019`

Programme :

- Dans le TD1, nous avons fait un rappel HTML/CSS ainsi qu'une introduction sur le **DOM (Document Object Model) et sa manipulation en JavaScript**. Nous avons également fait une introduction à la **programmation évènementielle**.
- Dans le TD2, nous avons vu comment **placer des images** relativement les une par rapport aux autres, ainsi que les **transformations 2D de similarité (translation, rotation, échelle)** et les coordonnées homogènes. Nous avons également vu deux **espaces de couleurs** parmi les espaces de couleurs existantes, à savoir RGB et HSL.
- Dans ce TD3, nous allons voir comment gérer **l'opacité/la transparence des images**. Nous allons également voir comment **créer et manipuler les images raster 2D**.

1 Transformations

Exercice 3.1 *Rappel HTML*

Dans cet exercice, vous allez établir un document HTML5 avec un placement absolu. Vous allez utiliser les transformations afin de placer un élément.

1. Dans votre dossier `in`, créer un dossier `td3`.
2. Avec Notepad++, créer un nouveau fichier `td2-transformations2D.html` dans ce dossier. Votre **arborescence de fichiers** sera donc :

```
in/
├── td1/
│   └── intro-javascript.html
├── td2/
│   └── td2-transformations2D.html
└── td3/
    └── td3-manipulation-images.html
```

3. Dans ce fichier, mettez les balises pour un document HTML5 vierge simpliste et enregistrez-le. Dans l'en-tête du document (`head`), déterminez le titre (balise `title`) "IN - TD3".
4. Dans le corps du document (après `body`), créez un titre `h1` "Imagerie numérique", et ensuite un titre `h2` "TD3 - Opacité/transparence et manipulation d'images".
5. Après le titre `h2`, ajoutez une balise paragraphe `p` avec le texte "L'objectif de ce TD est de :"
6. Après ce paragraphe, ajoutez une liste non-triée `ul`, avec trois éléments de liste avec le texte suivant.

- Comprendre comment gérer l'opacité et la transparence.
- Comprendre comment créer et manipuler des images 2D raster.
- Création et suppression des balises dans le DOM avec JavaScript.

Exercice 3.2 *Alpha compositing / alpha blending*

Dans cet exercice, nous allons voir comment gérer l'opacité/la transparence avec l'algorithme de peintre. Pour cela, en-sus des trois valeurs pour déterminer les composantes de couleurs (RGB ou HSL), nous allons spécifier une 4ème composante appelé α qui définit l'opacité et qui doit varier entre 0 et 1 (ou alors 0% et 100% ou bien 0 et 255. Une valeur au maximum définit un objet totalement opaque, et la valeur minimum définit un objet transparent.

Si on superpose un objet A sur un objet B, chacune des composantes de couleurs résultante C_o se calcule à partir des valeurs α_a et α_b et des composantes de couleurs C_a et C_b.

Une première façon de calculer est comme suit :

$$C_o = \frac{C_a\alpha_a + C_b\alpha_b\left(1-\alpha_a\right)}{\alpha_a + \alpha_b\left(1-\alpha_a\right)}$$

La valeur résultante α_o est :

$$\alpha_o = \alpha_a + \alpha_b\left(1-\alpha_a\right)$$

Une autre, deuxième façon de calculer est de prémultiplier les valeurs des composantes :

$$c_a = C_a\alpha_a$$

et ensuite calculer les valeurs résultante des composantes, et de α_o, d'une manière similaire :

$$c_o = c_a + c_b\left(1-\alpha_a\right)$$
$$\alpha_o = \alpha_a + \alpha_b\left(1-\alpha_a\right)$$

Par la suite, nous allons faire un exercice permettant de superposer un élément A sur un élément B.

1. Après votre liste, ajoutez un titre `h3` "Alpha blending - Gestion de l'opacité et transparence".
2. Insérez le code suivant ayant des balises `div` imbriquées.

```
<div class="B-fond" />
    <b>B</b>
    <div id="A100" class="superposition"><b>A</b> : opacité 1.0</div>
    <div id="A80" class="superposition"><b>A</b> : opacité 0.8</div>
</div>
```

3. En utilisant Notepad++, dans votre dossier `td3`, créez un sous-dossier `css` et dans ce dossier, un fichier `style.css`.
4. Dans votre fichier ajoutez le texte suivant :

```
h1, h2, h3
{
    color: rgb(0,0,0);
    background-color: rgb(221,221,221);
}
.B-fond
{
    position: relative;
    background-color: rgba(255, 0, 0, 1.0);
    width: 600px;
```

```
    height: 400px;
}
.superposition
{
    position: absolute;
    width: 150px;
    height: 30px;
    border: 1px solid black;
}
#A100
{
    transform: translate(25px, 50px);
    background-color: rgba(255, 255, 255, 1.0);
}
#A80
{
    transform: translate(25px, 100px);
    background-color: rgba(255, 255, 255, 0.8);
}
```

5. Ouvrez votre page web dans un navigateur.
6. Ajoutez 4 autres balises `div` pour A, avec les identifiants `A60`, `A40`, `A20`, et `A0`, à l'intérieur du `div` de B. Créez les sélecteurs CSS correspondants dans votre feuille de style, en faisant varier le canal alpha pour l'opacité de 0.6 à 0.0 par pas de 0.2.
7. Testez.
8. Calculez les valeurs des composantes pour chacune des couleurs résultantes en appliquant les formules ci-dessus.

	R_a	G_a	B_a	$alpha_a$	R_b	G_b	B_b	$alpha_b$	R_o	G_o	B_o	$alpha_o$
A100	255	255	255	1.0	255	0	0	1.0				
A80	255	255	255	0.8	255	0	0	1.0				
A60	255	255	255	0.6	255	0	0	1.0				
A40	255	255	255	0.4	255	0	0	1.0				
A20	255	255	255	0.2	255	0	0	1.0				
A0	255	255	255	0.0	255	0	0	1.0				

Exercice 3.3 *RGBa : espace de couleur combiné avec alpha blending*

1. Ajoutez un titre `h4` "Choix de couleur avec RGBa".
2. Tout comme la dernière fois, permettez de sélectionner une valeur RGBa interactivement. Pour cela, prenez votre sélectionneur de couleur RGB, et ajoutez y un canal alpha entre 0 et 100%, que vous normalisez ensuite entre 0.0 et 1.0. Par défaut, mettez le canal alpha à 100%.

qq

2 Images 2D matricielles

Par la suite, nous allons préparer un logiciel de dessin 2D dans votre navigateur, qui permet à vos utilisateur de dessiner avec un pinceau, et de déterminer la forme et taille du pinceau ainsi que sa couleur. A la fin, nous appliquons l'image généré et nous la superposons sur le fond B de l'exercice précédente.

Exercice 3.4 *Images 2D matricielles dans la balise Canvas*

Dans cet exercice, nous allons découvrir la balise `canvas`. Cette balise nous permet de créer des images 2D matricielles.

1. Dans votre fichier HTML, créez un nouveau titre `h3` "Images 2D matricielles dans un canvas".

```
<canvas id="image2D" width="600" height="400"></canvas>
<br />
<button id="boutonEffacer">Effacer</button>
<div id="couleurActuel">
</div>
```

2. Dans votre fichier CSS, ajoutez les sélecteurs suivants :

```
#couleurActuel
{
    width:50px;
    height: 50px;
    background-color: rgba(255,0,0,1.0);
}

#image2D
{
    border: 1px solid rgb(128,128,128);
}
```

3. Testez le résultat.
4. Ensuite, ajoutez le script JavaScript suivant dans votre fichier HTML :

```
let imageCanvas = document.getElementById("image2D");

let imageContexte = imageCanvas.getContext("2d");
let imageDonnees = imageContexte.getImageData(0, 0, imageCanvas.width, imageCanvas.height);

console.log(imageDonnees.data);
// vos instructions ici

imageContexte.putImageData(imageDonnees,0,0);
```

5. Analysez la sortie de votre console. Quelle est la taille du tableau `imageDonnees.data` ? Savez vous pourquoi ?
6. A l'endroit indiqué dans le script précédent, ajoutez les instructions suivantes :

```
imageDonnees.data[12] = 0;
imageDonnees.data[13] = 0;
imageDonnees.data[14] = 255;
imageDonnees.data[15] = 255;

console.log(imageDonnees.data);
```

7. Testez. Voyez vous le nouveau pixel dans votre image ? A quel position est-ce qu'il est ?
8. Créez un nouveau pixel de couleur rouge au milieu de votre canvas. Réfléchissez notamment à la manière de trouver les bons indices dans le tableau.

Vous pouvez également définir une fonction pour trouver le bon index du tableau :

```
function calculerIndex(x, y, largeur)
{
    return x*4 + (y * largeur * 4);
}
```

Pour utiliser cette fonction pour un pixel à la position (20, 30), vous pouvez l'appeler et affectez les canaux de couleurs correspondants :

```
index = calculerIndex(20, 30, imageCanvas.width);
imageDonnees.data[index] = 0;
imageDonnees.data[index+1] = 0;
imageDonnees.data[index+2] = 255;
imageDonnees.data[index+3] = 255;
```

Exercice 3.5 *Créez une ligne dans une image matricielle*

1. Affectez maintenant tous les pixels de la ligne 200 avec une couleur grise (p.ex. 127).
2. Affectez maintenant tous les pixels dans le rectangle défini par les coins en haut à gauche coordonnées [10,10], et le coin plus bas et plus à droite [40,200], rempli avec la couleur bleue. Utilisez des boucles imbriquées (ici, une boucle dans une boucle).

Notez qu'en JavaScript, vous avez plusieurs façon de faire des boucles :

- Boucle de type `for`

```
for (let i=0; i<10; i++)
{
    console.log(i);
}
```

- Boucle de type `while`

```
let i=0;
while (i<10)
{
    console.log(i);
    i=i+1;
}
```

3. Sauvegardez l'image crée sous le nom td3.png en utilisant le clic droit dans votre navigateur. Ouvrez l'image générée avec un logiciel d'édition d'images.
4. Créez une fonction permettant de dessiner un rectangle dans un canvas avec une couleur donnée.

```
function rectangle(canvas, x, y, largeur, hauteur, r, g, b, a)
{
    ....
}
```

Notez que vous pouvez vous en servir de votre fonction `calculerIndex`.
5. Testez votre fonction en créent différents rectangles.

Exercice 3.6 *Fonctions de callback : boutons*

1. Faites en sorte qu'en cliquant sur le bouton "Effacer" que le canvas devienne de nouveau tout blanc. Inspirez vous d'associer une fonction de *Callback*, tout comme dans l'exercice 1.16 du TD 1.
2. Ajoutez un nouveau bouton "Remplir avec couleur actuel", et associez une nouvelle fonction qui remplisse le canvas avec la couleur actuel. Notez : déclarez des variables globales `actuelR`, `actuelG`, `actuelB`, et `actuelA` que vous mettez à jour en manipulant vos sliders.

Exercice 3.7 *Fonctions de callback : évènement de la souris*

1. Au-dessus de votre balise `canvas`, ajoutez les balises suivantes :

```
<i>Contenu : Manipulations de pixels, fonctions de callback</i>
<br />
<i>x:</i> <b id="coordX"></b>
<br />
<i>y:</i> <b id="coordY"></b>
<br />
<i>rgba:</i> <b id="rgba"></b>
<br />
```

```
imageCanvas.addEventListener("mousemove", montrerCoordonnees)

function montrerCoordonnees(event)
{
    let x = event.offsetX;
    let y = event.offsetY;
    document.getElementById("coordX").innerHTML = x;
    document.getElementById("coordY").innerHTML = y;

    let p = imageContexte.getImageData(x, y, 1, 1).data;

    document.getElementById("rgba").innerHTML = "rgb("+p[0]+","+p[1]+","+p[2]+")";
}
```

2. Changez votre script pour que vous dessinez un pixel avec couleur rouge à l'endroit du pointeur de la souris.
3. Au lieu d'une couleur rouge, prenez la couleur actuelle.
4. Dans des logiciel d'édition d'images, il est courant d'utiliser des brosses (ou pinceaux) de différentes formes et tailles, donc d'affecter plusieurs pixels à la fois. Créez donc un carré rempli de la couleur rouge aux alentours du pixel sélectionné. Comme taille, chosissez par exemple 5.
5. Définissez un *slider* variant de 1 à 10 afin de définir la taille de la brosse.

Exercice 3.8 *Ajouter des balises en JavaScript*

Dans cet exercice, nous allons apprendre comment ajouter des balises en utilisant JavaScript.
1. Dans votre fichier `td3-manipulation-images.html`, après la balise canvas, ajoutez les balises suivantes permettant de définir le début d'une palette de couleur :

```
<i>Votre palette de couleur :</i>
<div id="conteneurPalette">
    <div class="paletteCouleur" id="paletteNoir"></div>
    <div class="paletteCouleur" id="paletteBlanc"></div>
</div>
```

2. Dans votre fichier CSS, ajoutez les sélecteurs suivants :

```
.conteneurPalette
{
    display: inline;
}
.paletteCouleur
{
    width:50px;
    height: 50px;
```

```
    border: 1px solid rgb(128,128,128);
    margin: 2px;
    display: inline-block;
}

#paletteBlanc
{
    background-color: rgba(255,255,255,255);
}
#paletteNoir
{
    background-color: rgba(0,0,0,255);
}
```

Ouvrez votre page dans un navigateur.

3. A côte de votre sélectionneur de couleur RGBA avec les sliders, ajoutez un bouton "Ajouter à la palette" et définissez une fonction de *Callback* permettant d'appeler la fonction suivante qui créé une nouvelle balise de la classe `paletteCouleur` et qui l'ajoute dans le conteneur de la palette. Pour l'instant, la couleur est rouge.

```
function ajouterPalette()
{
    let nouveauDiv = document.createElement('div');
    nouveauDiv.classList.add("paletteCouleur");
    nouveauDiv.style.backgroundColor = "rgba(255,0,0,255);

    let conteneurPalette = document.getElementById("conteneurPalette");
    conteneurPalette.appendChild(nouveauDiv);
}
```

4. Avec vos outils de développement du navigateur, inspectez (examinez) l'arbre DOM et vérifiez l'endroit des balises ajoutées.
5. Modifiez pour que la nouvelle couleur de la palette sera de la couleur choisie dans votre sélectionneur de couleur.

Exercice 3.9 *Dessiner un graphe d'une fonction*

Dans cet exercice, on essayera de tracer la fonction univariée suivante dans votre canvas :

$$f(x) = 0.004x^2 + 0.004x + 30$$

1. Définir d'abord une fonction originale `f(x)` pour cette fonction.
2. Dessiner cette fonction dans votre canvas pour des valeurs de x variant de 0 à 300;

Exercice 3.10 *Dessiner un cercle*

1. Dans votre canvas, dessiner un cercle remplie avec la couleur verte ayant comme centre le pixel (200x200) avec un rayon de 150 pixels.
2. Comme pour le rectangle, définissez une fonction.
3. Combien de différentes methodes pour déssiner un cercle connaissez-vous ?