TP - API Canvas

1 Une zone de dessin dans vos pages web

Lors de vos recherches de balises HTML, vous avez peut-être déjà croisé une balise nommée **<canvas>**. Celle-ci sert de support pour **dessiner différentes formes géométriques, mais aussi charger des images**. C'est notamment grâce à cette balise que la plupart des **jeux vidéo créés en Javascript** prennent vie.

1.1 Quel intérêt pour nous ?

Mis-à-part pour créer des animations, simulations et autres jeux vidéo, effectivement cette balise n'a pas vraiment sa place sur un site internet. De plus, nous n'allons rien faire de tout cela... Alors quoi ? **Nous exercer à comprendre l'algorithmique !** L'exécution d'algorithmes n'est pas toujours simple à comprendre. Passer par un support visuel pourra peut-être vous aider à mieux saisir les subtilités de l'algorithmique. Nous allons donc aujourd'hui tenter de **dessiner des motifs**, et autres formes géométriques à l'aide de **boucles et de conditions**.

1.2 Comment ça marche ?

La balise <canvas> peut prendre en paramètre une **hauteur et une largeur**, qui définiront la taille de la zone de dessin. Il s'agira ensuite de **récupérer un "contexte"** de ce canvas en Javascript qui sera l'objet nous permettant de dessiner. Il existe différents contextes, les deux principaux étant le **contexte 2D** et le contexte webgl pour faire de la 3D. Nous allons nous contenter du contexte 2D pour ce TP. Une fois le contexte récupéré, nous allons appeler des fonctions depuis celui-ci afin de dessiner les formes géométriques que l'on veut.

Afin de vous familiariser avec l'API, un tutoriel est disponible ici `https://developer.mozilla.org/fr/docs/Web/API/Canvas_API/Tutoriel_canvas`. N'hésitez pas à **explorer la documentation**, expérimenter des choses, et trouver des ressources au-delà de ce que ce TP vous fournit.

2 Objectif du TP

Vous trouverez sur École Directe les sources du TP d'aujourd'hui. Il s'agit d'une page html avec un peu de code. Vous trouverez deux Canvas et quinze boutons. Chaque bouton correspond à un dessin qui s'affichera sur le cadre de droite. Il vous faudra **reproduire à l'identique** ces dessins sur la zone de gauche.

- **Si vous avez Node.js sur votre machine**, vous pourrez lancer le fichier **src/index.html** avec parcel. Modifiez **src/drawings.ts** afin de compléter vos dessins et compilez le projet avec la commande **"tsc"**.
- **Si vous n'avez pas Node.js**, utilisez **Wamp** (ou alternative) pour héberger votre projet. Copiez-collez le dossier **TP_canvas** dans la racine (www chez Wamp), et lancez une page web sur `"localhost:80/TP_canvas"`. Vous devrez modifier le fichier **src/dist.browser/drawings.js** pour faire apparaître vos dessins.

Chaque fonction du fichier drawings.ts (ou drawings.js) prend **deux paramètres**, le **canvas** et le **contexte** permettant de dessiner dans le cadre de gauche lorsque vous cliquerez sur le bouton correspondant.

Vous travaillerez **seuls** pour la réalisation de ce TP, il n'est néanmoins **pas interdit de s'entraider**.

2.1 Rendu de TP

Le rendu se fera sous forme d'**un fichier drawings.ts (ou drawings.js) dûment complété**. Vous avez jusqu'au **vendredi 18 décembre**, fin de séance (10h ou 12h selon le groupe) pour déposer votre fichier dans **la zone travail à rendre** de la séance.

3 À vos crayons !

3.1 Dessine-moi un carré

3.1.1 L'objectif

Le premier exercice consiste à dessiner **trois carrés de couleurs différentes** : rouge, vert, bleu. Ces 3 carrés sont de **50px de côté**, ils sont espacés de **10px** du bord haut, du bord gauche, et entre eux.

3.1.2 Les nouvelles notions

Vous devrez utiliser **context.fillStyle** afin de choisir la bonne couleur à appliquer **avant chaque dessin** de rectangle. Vous utiliserez ensuite **context.fillRect()** afin de **dessiner le rectangle** correspondant à la couleur choisie.

3.2 Voyons plus grand

3.2.1 L'objectif

Cette fois nous allons dessiner ces 3 mêmes rectangles, mais ils vont prendre un peu plus de place. Les rectangles sont espacés de 10px **de chaque bord, et les uns des autres**.

3.2.2 Les nouvelles notions

Ici nous aurons besoin de **connaître la hauteur et la largeur** du cadre dans lequel nous dessinons. L'accès à ces informations se fait **via le canvas**, avec **canvas.width** et **canvas.height**.

3.3 On renverse tout !

3.3.1 L'objectif

Nous allons reprendre une dernière fois nos rectangles, mais cette fois les étendre **horizontalement**. Les rectangles sont espacés de 10px **de chaque bord, et les uns des autres**.

3.4 Changeons de forme

3.4.1 L'objectif

Marre des rectangles ? Nous allons maintenant dessiner des **cercles**. Ces cercles seront **un peu transparents (un alpha de 0.5)** et se superposeront. Ils seront disposés **selon un triangle équilatéral de côté 150**. Magnanime, je vous donne les coordonnées des points a, b, c du triangle.

3.4.2 Les nouvelles notions

Nous allons cette fois **ajouter le canal alpha** à notre rgb classique. Il va également falloir maîtriser les fonctions **context.beginPath()**, **context.arc()** et **context.fill()**.

3.5 Force, Sagesse, Courage

3.5.1 l'objectif

Continuons sur notre lancée et dessinons quelque chose de plus concret. Nous allons garder la forme de triangle de l'exercice précédent, pour cette fois réellement **dessiner des triangles**. Notre triangle équilatéral sera **de côté 300**. Il faudra **calculer les coordonnées manquantes pour compléter le dessin**.

3.5.2 Les nouvelles notions

Première information à noter, **context.fillStyle prend aussi des couleurs en code hexadécimal**. Nous allons réutiliser **context.beginPath()**. Cette fois néanmoins, pas d'arc, mais des lignes. La fonction **context.moveTo()** et la fonction **context.lineTo()** nous permettront de tracer tout ce qu'il nous faut. N'oubliez pas de compléter vos formes avec **context.fill()** pour donner vie à votre dessin.

3.6 La tapisserie de mamie

3.6.1 L'objectif

L'échauffement est terminé... Il est temps d'attaquer **l'algorithmique**, la vraie, celle avec **des conditions et des boucles**. Nous allons repartir sur nos rectangles de début de TP, cette fois, pas question de tous les dessiner à la main. Nous allons **utiliser une boucle afin de générer chaque rectangle**. Un rectangle sur 2 aura une couleur claire, et l'autre une couleur plus foncée.

3.6.2 Les nouvelles notions

Il est temps de générer du rectangle à la boucle. Il faudra habilement **exploiter notre variable d'index i** afin de parvenir à nos fins. Afin de générer **une étape sur deux** avec la couleur correspondante, il serait utile de **comprendre et maîtriser l'opérateur modulo (%)**.

3.7 Un chouette dégradé

3.7.1 L'objectif

Gardons notre boucle, mais cette fois, nous allons faire **un dégradé de couleurs** en utilisant chaque rectangle. Je vous donne **la couleur de départ et la couleur de fin dans deux tableaux**. Je calcule également **le pas du dégradé de chaque composante rgb**. Il s'agit maintenant d'utiliser ces données correctement afin de dégrader la couleur sur chaque rectangle.

3.7.2 Les nouvelles notions

Nous allons **calculer dynamiquement les couleurs** que nous manipulons. Il va falloir **créer une chaîne de caractères "rgb(x, y, z)"** pour faire varier la couleur que nous donnons au context.fillStyle.

3.8 Encore du dégradé

3.8.1 L'objectif

Nous allons cette fois générer un dégradé de rectangles **qui s'étend dans les deux directions**. Pas de taille de rectangle, mais un nombre total de rectangles à afficher dans le canvas.

3.9 Toujours du dégradé

3.9.1 L'objectif

Cette fois nous allons générer **un cercle de dégradé**. Toujours les mêmes couleurs de départ et d'arrivée. Il va falloir bien comprendre les mécanismes du tracé manuel de forme et des arcs de cercle.

3.10 Le secret de leur pouvoir

3.10.1 L'objectif

C'est l'heure de la récré. On se calme avec les boucles et les dégradés. À l'aide des variables de départ, il va falloir dessiner ce bijou de technologie, le nec plus ultra de la prison portative pour animaux mignons.

3.10.2 Les nouvelles notions

Une nouveauté afin de dessiner les contours : la fonction **context.stroke()** viendra remplacer le traditionnel **context.fill()**.

3.11 Le torchon de mamie

3.11.1 L'objectif

On va monter à nouveau d'un cran, et retrouver notre amie à tous : la **double boucle imbriquée**. À l'aide des variables de départ, générer ce damier.

3.12 Palette de couleurs

Le dégradé vous avait manqué ? Moi oui ! Nous allons **appliquer le dégradé à notre double boucle imbriquée** "tout simplement".

3.13 Art abstrait

3.13.1 L'objectif

Nous allons toujours plus loin dans la double boucle, mais cette fois nous allons **dessiner des arcs de cercle**. La migraine commence à s'installer...

3.14 Le carrelage de mamie

3.14.1 L'objectif

Ce dernier dessin peut être résolu de plusieurs manières. Il demandera néanmoins une petite dose de réflexion, et sûrement plusieurs tentatives.

3.15 L'inspiration du maître

3.15.1 L'objectif

Vous êtes à présent **des chefs du dessin et de l'algorithmique**. Il est temps de **faire votre propre dessin**, trouvez l'inspiration et gratifiez-moi de **votre plus belle oeuvre**.