Compte rendu : *Projet Virtual Landscape*

Tables des matières

Projet réalisé par :
Belkacem Nassim

Du 05/04/2023 au 14/05/2023

Contexte :

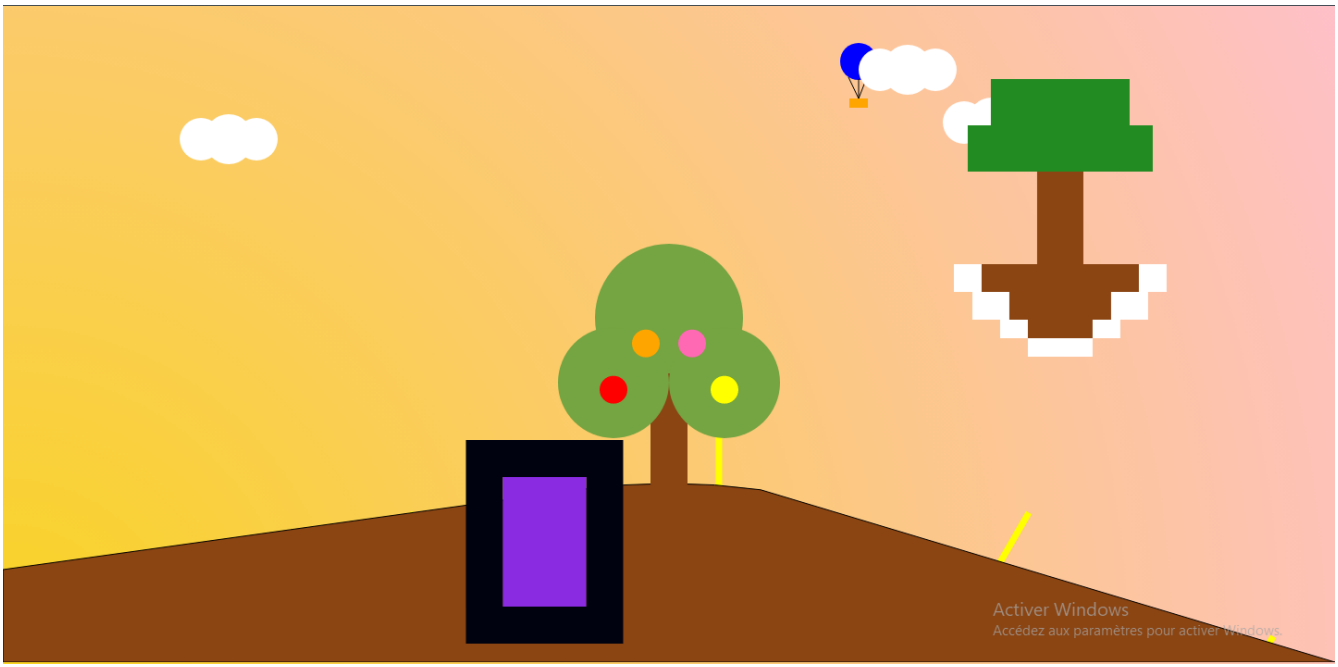
```
Dans le contexte du projet Virtual Landscape, notre objectif était d'acquérir de
nouvelles compétences en programmation avec le langage JavaScript, en adoptant une
approche orientée objet et événementielle. Nous avons également cherché à explorer une
API de manière autodidacte et à développer notre créativité dans le processus.
```

Introduction :

```
Une fois les objectifs du projet établis, nous aborderons la création et l'utilisation
d'objets, ainsi que les défis rencontrés tout au long du processus. Enfin, nous
conclurons avec une réflexion personnelle.
```

I. Création du projet

Je me suis inspiré de l'univers de Minecraft pour créer ce paysage ainsi que d'un désert. Les rectangles créant des formes rappelant les possibilités infinies de Minecraft, comme l'ile volante, ou alors le portail du Nether (une autre dimension) rappelant directement son univers.



Nous allons maintenant expliquer l'objet "Nuage", en gardant à l'esprit que la structure du code est la même pour tous les autres objets, à l'exception des valeurs spécifiques et du dessin tracé.

Tout d'abord, nous créons un nouveau fichier que l'on nommera Nuage.js.

Le fichier de code présent importe la classe "AbstractForm" depuis le fichier "AbstractForm.js", ce qui permet à la classe "Nuage" d'en hériter et de fonctionner grâce à ses propriétés et méthodes.

```
{ AbstractForm } from './AbstractForm.js';
```

La classe "Nuage" possède un constructeur qui accepte neuf paramètres avec des valeurs spécifiques. Ces paramètres incluent les coordonnées x et y, la largeur et la hauteur, les couleurs de remplissage et de contour, l'épaisseur du contour, la pesanteur et l'ordre de construction.

```
export class Nuage extends AbstractForm {
  // add default values to avoid errors on empty arguments
  constructor(
    x = 0,
    y = 0,
    width = 0,
    height = 0,
    fillColor = '',
    strokeColor = '',
    strokeWidth = 2,
    pesenteur = false,
    ordreConstruction = 100
  ) {
    super(x, y, width, height, fillColor, strokeColor, strokeWidth, pesenteur,
ordreConstruction);
  }
```

Ce code invoque le constructeur de la classe "AbstractForm" en lui transmettant les paramètres suivants. Cela permet à la classe "Nuage" de bénéficier de toutes les fonctionnalités de la classe "AbstractForm", tout en conservant ses propres paramètres distincts.

```
{
  super(x, y, width, length, fillColor, strokeColor, strokeWidth, pesenteur,
ordreConstruction)
}
```

Nous créons la méthode appelée nuage. Nous stockons les coordonnées dx et dy dans les variables ox et oy. Puis nous dessinons le coquillage avec les variables ox et oy.

```
  nuage(ctx, dx, dy) {
    let ox = dx;
    let oy = dy;

    // nuage (milieu)
    ctx.beginPath();
    ctx.arc(ox + 210, oy + 80, 27, 0, Math.PI * 2, true);
    ctx.fillStyle = 'white';
    ctx.fill();

    // nuage (coté gauche)
    ctx.beginPath();
    ctx.arc(ox + 180, oy + 80, 23, 0, Math.PI * 2, true);
    ctx.fillStyle = 'white';
    ctx.fill();

    // nuage(coté droit)
    ctx.beginPath();
    ctx.arc(ox + 240, oy + 80, 23, 0, Math.PI * 2, true);
    ctx.fillStyle = 'white';
    ctx.fill();
  }
```

La méthode draw dessine l'objet Nuage en appelant la méthode nuage. Elle sauvegarde les "ctx" puis elle appelle la méthode nuage pour dessiner l'objet. Enfin, elle restaure les ctx antérieurs.

```
draw(ctx){
 ctx.save()
 this.Nuage(ctx, this.x, this.y)
 ctx.restore()
}
```

La ligne de code indique que la fonction retourne un tableau d'objets.

```
@return {[Nuage,...]}
```

La méthode construit un tableau d'instances de la classe Nuage. Nous utilisons une boucle for pour créer un nombre aléatoire d'instances de Nuage. Nous reprenons les mêmes paramètres cités précédemment.

```
  /**
   * get array of forms
   * @return {[Nuage,...]}
   */
  static buildForms() {
    // create a new nuage object using the Nuage class
    let forms = []
    for (var i = 0; i <= ~~(Math.random() * 6); i++) {
      forms.push(new Nuage(~~(Math.random() * 1000), ~~(Math.random() * 150) - 50,
100, 100, 'blue', 'black', 1, true, 3))
    }

    console.log('nb de nuage : ' + forms.length)

    // retourne un tableau d'objets de type Nuage
    return forms
```

NOTE	Si nous voulons un nombre précis de Nuages, nous pouvons nous dispenser d'une boucle. De plus, nous pouvons aussi nous dispenser de "Math.ramdom()" si nous souhaitons une coordonnée (x et y) fixe.

Enfin, la méthode retourne le tableau de toutes les Nuages créées.

```
return forms
```

Et nous exportons la classe avec la ligne de code ci-dessous afin de visualiser nos nuages.

```
export { Nuage } from './Nuage.js';
```

II. Difficultés rencontrées :

L'un des premiers défis, courant dans tout nouveau projet, a été de comprendre les objectifs et d'acquérir de nouvelles connaissances.

Ensuite, j'ai rencontré des difficultés pour trouver l'inspiration. Il m'a été ardu de concevoir un projet original tout en veillant à ce qu'il reste réalisable et qu'il puisse être livré dans les délais impartis.

Enfin, j'ai également fait face à des difficultés liées à l'utilisation de certaines méthodes.

III. Conclusion :

Ce projet a été une expérience très instructive qui m’a permis d’adopter une approche de programmation différente. Cependant, j’ai eu du mal à intégrer et à appliquer certains concepts fondamentaux tels que la gravité, qui se sont avérés essentiels pour mon projet. De plus, j’ai ressenti des limites, car mes objets se résumaient souvent à des combinaisons de traits et de cercles, ce qui m’a parfois contraint à recourir à des ressources externes (forums, vidéos YouTube, ChatGPT) pour créer des objets originaux.