DGEP

Rapport Snake

Tiago Rodrigues Sousa
09/01/2024

Table des matières

1. Introduction

Ce rapport décrit le projet Snake qui a été fait lors du cadre du projet P-Bulles. Le but était de faire un snake en JS pure. Lors du commencement du projet, il a été nécessaire d'apprendre les bases du JS.

2. Analyse

Avant de se lancer sur le programme, une analyse a été faite sur le fonctionnement du snake. La réflexion a été faite sur comment le snake est fait et comment il doit bouger.

Pour la partie comment il est fait, il a été décidé que chaque carré représentait une partie su serpent et que toutes les parties du snake serait dans un tableau.

Pour la partie mouvement du snake, il a été décidé qu'on supprime la dernière partie du snake et qu'on rajoute une partie au début (sa tête).

3. Réalisation

3.1. Template

Lors du commencement, il nous a été fourni un canva avec une fonction move (fonction principale).

3.2. Comment est définit la taille du serpent

La variable tailleSerpent va définir la taille de chaque bloc en pixel. On va utiliser cette constante pour que cela reste proportionnel selon la taille choisit.

```
const tailleSerpent = 50;
```

3.3. Création du serpent et des pommes

Lorsque le jeu commence, on va créer un objet snake qui va contenir les parties partofsnake et les cordonnées x et y de la partie tout devant (la tête).

```
let Snake = {
  part: [],
  x:tailleSerpent* 2,        //x de la part qui est devant
  y:0,                       //y de la part qui est devant
}
```

```
Snake.part.push(new PartOfSnake(0, 0), new PartOfSnake(tailleSerpent, 0), new
PartOfSnake(tailleSerpent*2, 0));
```

Puis un objet Apple va être créer et on va lui donner des cordonnées aléatoire.

```
//nouvelle pomme
function newApple(){
  let numberX;
  let numberY;
  let ok = true;

  //contrôle si la pomme n'apparaît pas sur le serpent
  do
  {
    ok = true                                                    //varibale
pour savoir si les nombré généré
    numberX = Math.floor(800/tailleSerpent * Math.random())
    numberY = Math.floor(800/tailleSerpent * Math.random())

    if (Snake.part.some((n1) => (n1.getX() == numberX*tailleSerpent &&
numberY*tailleSerpent == n1.getY())))
    {
      ok=false;
    }
  }
  while(!ok)

  Apple.x = numberX*tailleSerpent
  Apple.y = numberY*tailleSerpent
}
```

On va directement faire bouger le snake à droite grâce au UserInputX.

3.4. Programme principal

Dans le programme principal (move) on va d'abord définir la zone de jeu, son emplacement et sa taille.

```
//programme principale
const move = () => {
  // Dessine la grille de jeu
  ctx.fillStyle = 'black';
  ctx.fillRect(0, 0, 800, 800);

  //va bouger le serpent
  if (frame == 30){
    //s'il meurt
    death();

    //mange la pomme ou mouvement
    Snake.x + userInputX == Apple.x && Snake.y + userInputY == Apple.y ?
addAppleInPart()
    : mouvementSnake(Snake.x + userInputX, Snake.y + userInputY);

    Snake.x += userInputX;
    Snake.y += userInputY;

    //dessine le serpent
    frame = 0;
  }
```

3.5. Contrôle si mort

Puis toutes les 30 frames (nombre de fois qu'il fait la boucle), on va vérifier s'il meurt grâce à une fonction (death). Cette fonction va regarder si le serpent sort de la zone de jeu ou s'il se touche lui-même, si c'est le cas alors l'écran de fin se lance, sinon il ne se passe rien.

```
//contrôle si mort
function death(){
  //s'il meurt
  if (Snake.x + userInputX < 0 || Snake.x + userInputX > 800 - tailleSerpent
|| Snake.y + userInputY < 0 || Snake.y + userInputY > 800 - tailleSerpent ||
      Snake.part.some((n1) => (n1.getX() == Snake.x + userInputX && Snake.y +
userInputY == n1.getY())))
  {
    //fait disparaitre écran jeu et fait apparaitre game over
    document.getElementById('app').style.display = 'none';
    document.getElementById('gameOver').style.display = 'block';

    //affiche différent écran
    if(firstTime)
    {
      firstTime = false;
      document.getElementById('gameOver').innerHTML = "Vous etes mort!";
```

```
    }

    setTimeout(() => {
      document.getElementById('gameOver').innerHTML = "Score : " + score;
    }, 2000);

    setTimeout(() => {
      document.location.href="index.html"
    }, 4000);
  }
}
```

3.6. S'il mange une pomme

Ensuite on regarde s'il mange une pomme ou bien s'il avance. S'il mange une pomme alors on va dans une fonction (addAppleInPart) qui va ajouter une partie à la fin du tableau (Snake.part) qui correspond à la tête du serpent. Puis on relance les cordonnées de les pomme (newApple).

```
//lorsque le joueur mange une pomme
function addAppleInPart(){
  Snake.part.push(new PartOfSnake(Apple.x, Apple.y));
  newApple();
  score++;
}
```

3.7. S'il bouge

S'il ne mange pas de pomme alors il va bouger (mouvementSnake). Pour ça, on retire la dernière partie du serpent et on lui rajoute une nouvelle tête grâce au cordonné.

```
//fait bouger le serpent
function mouvementSnake(x, y){
  Snake.part.shift();
  Snake.part.push(new PartOfSnake(x, y));
}
```

Le calcul des cordonnées du serpent

Pour le calcul des cordonné on va utiliser les cordonnées de la tête du serpent + la valeur de l'input du user.

Ensuite on définit dans l'objet snake, les cordonnées de la nouvelle tête.

3.8. Affichage

Puis on dessine les différentes parties et la pomme (Draw, drawApple). Pour l'affichage du serpent, on utilise un forEach qui va afficher les parties du snake une par une. Puis on affiche le score.

```
//dessine la pomme
function drawApple(){
  ctx.fillStyle = 'red'
  ctx.fillRect(Apple.x, Apple.y, tailleSerpent, tailleSerpent)
}
```

```
function Draw(){
  ctx.fillStyle = couleurSnake;
  Snake.part.forEach((element) => {
    ctx.fillRect(element.getX(), element.getY(), tailleSerpent, tailleSerpent)
  });
}
```

4. Conclusion

Ce projet a permis d'apprendre le JS de manière ludique. Il a été difficile de commencer car même avec de la théorie, ça été compliqué d'imaginer comment faire. Il aurait été préférable que je fasse une analyse pour savoir comment je vais coder.

Après avoir compris comment marche le Canvas, la suite était plus facile. Cela s'est bien déroulé, et à part des problèmes de syntaxe dû à la flexibilité de JS, aucun gros problème.

Pour le mémo, seules les nouvelles choses ont été notées.

Tous les critères demandés ont été intégrer. Les pistes d'amélioration pour le projet serait un système multijoueur qui permettrait de montrer l'utilité des objets.

En résumé le projet c'est bien passé et les compétences demandées ont été acquises.