

Snake JS (Code à trous)

"SSSSSSSS'est que je commenSSSSSSSe à avoir faim moi" se dit Snaky, votre serpent de compagnie favoris. Et oui, un serpent aussi ça a besoin de manger ! Mais comme son maître n'est pas à la maison, il décide d'aller chercher sa nourriture lui même.

Parviendrez-vous à l'aider dans cette tâche ?

Consignes

Pour pas perdre des plumes

A vous de jouer !

Accéder au projet

Il est pas beau mon serpent ?

Mi casa es tu casa

Conclusion

Pour aller plus loin

Consignes

- Si vous avez des questions, pensez à demander de l'aide à votre voisin de droite. Puis de gauche. Demandez enfin à un Cobra (ceux-là ne mordent pas) si vous êtes toujours bloqué(e).
- Vous avez tout à fait le droit d'utiliser internet pour trouver des réponses ou pour vous renseigner.



- N'hésitez pas à faire des bonus et à ajouter des fonctionnalités lorsque votre projet sera terminé et validé.

Pour pas perdre des plumes

Avant d'aller plus loin, vous trouverez ici une liste de fonctions et d'aides qui vous seront utiles pour terminer ce projet. N'hésitez pas à revenir jeter un coup d'œil si vous avez des questions.

▼ Fonctions utiles

Fonctions

 Prototype	 Description
<u>floor(nb)</u>	Arrondi la valeur <code>nb</code> à l'inférieur. <u>Exemples</u> : <code>floor(5.95)</code> → 5, <code>floor(-5.05)</code> → -6, <code>floor(1/2)</code> → 0
<u>random(nb)</u>	Vous donne un nombre aléatoire entre 0 et <code>nb</code> . Attention, ce nombre peut être à virgule.

A vous de jouer !

Accéder au projet

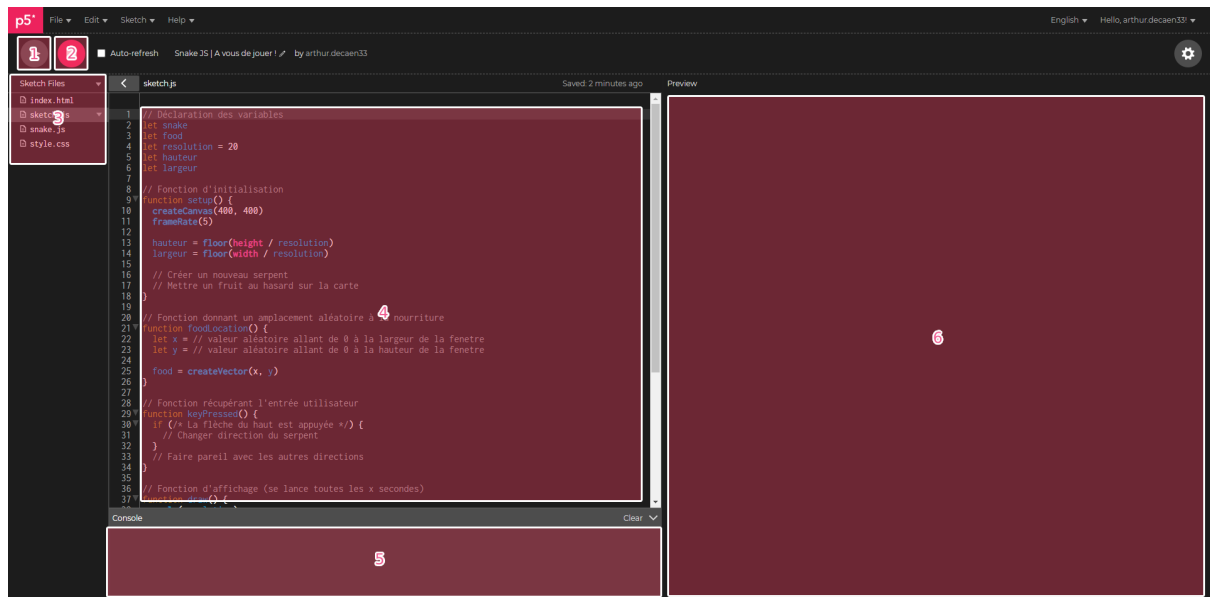
Le projet est accessible depuis la lien ci-après. Vous pouvez vous créer un compte sur le site si vous souhaitez garder une trace de ce que vous avez fait. Dans le cas contraire, vous pouvez simplement commencer à coder !

p5.js Web Editor

A web editor for p5.js, a JavaScript library with the goal of making coding accessible to artists, designers, educators, and beginners.

★ https://editor.p5js.org/arthur.decaen33/sketches/GojI_g7R9

Et voici donc l'interface sur laquelle vous allez travailler :



1. Lancer votre programme
2. Arrêter votre programme
3. Fichiers de votre projet
4. Code du fichier ouvert
5. Console, permet d'afficher les erreurs ou les textes de votre programme
6. Zone d'affichage de votre jeu

Le projet que vous avez entre les mains est un projet "à trous". C'est à vous de faire en sorte que tout marche à l'aide des commentaires présents. Vous trouverez également des indications supplémentaires dans la suite de ce sujet.

Il est pas beau mon serpent ?

Pour commencer l'aventure de Snaky le serpent au travers de la maison sauvage, il va falloir rendre Snaky intelligent.

Pour cela, rendez-vous dans le fichier `snake.js`.

▼ Dissection d'un serpent (pas un vrai je vous rassure)

Snake est un objet, et comme chaque objet il possède des attributs (ses variables) et ses fonctionnalités (ses méthodes).

Les variables de Snaky sont données dans la partie `constructor` du fichier. Dans notre cas, il possède un corps (`body`), une position de départ (`body[0]`), une

vitesse verticale (`ydir`), une vitesse horizontale (`xdir`) et une longueur (`len`).

Pour se qui est des méthodes, notre serpent peut se montrer (`show`), se mettre à jour (`update`), changer de direction (`setDir`), grandir (`grow`), manger (`eat`) et perdre (`endGame`).

Pour rendre notre serpent intelligent, il va donc falloir programmer ses méthodes. Pour vous aider, elles sont à trous, à vous de les compléter. Vous pouvez cliquer sur le nom des méthodes ci-après pour avoir des indices (et pourquoi pas la solution) :

▼ `update()`

Cette fonction s'exécutera à chaque frame de votre jeu, c'est elle qui affecte le mouvement du serpent à ses différents morceaux.

Le but ici est d'ajouter à `head.x` et `head.y` la direction qui est appliqué au serpent.

▼ Indice 1

Pour utiliser une variable se trouvant dans le serpent, il faut l'écrire de la manière suivante : `this.ma_variable` .

▼ Indice 2

Les variables à utiliser ici sont les variables `xdir` et `ydir` .

▼ **Solution**

```
update() {  
  let head = this.body[this.len - 1].copy();  
  this.body.shift();  
  head.x += this.xdir;  
  head.y += this.ydir;  
  this.body.push(head);  
}
```

▼ `show()`

Cette fonction va afficher chaque partie du corps du serpent à la position où elle doit se trouver à l'écran.

▼ Indice 1

Le corps du serpent est dans un tableau nommé `body` .

▼ Indice 2

Pour accéder à une case d'un tableau, il faut l'écrire de la manière suivante : `mon_tableau[index]` . Vous n'auriez pas une variable qui peut servir d'index ?

▼ Solution

```
show() {  
  for (let i = 0; i != this.len; i++) {  
    fill(0);  
    noStroke();  
    rect(this.body[i].x, this.body[i].y, 1, 1);  
  }  
}
```

▼ setDir(x, y)

Le but de cette fonction est de modifier la direction du serpent.

▼ Indice 1

Regardez les variables du serpent dans la partie `constructor` pour voir si une ou plusieurs correspondent à ce que vous cherchez.

▼ Indice 2

Les variables à modifier sont `xdir` et `ydir` .

▼ Solution

```
setDir(x, y) {  
  this.xdir = x;  
  this.ydir = y;  
}
```

▼ grow()

Cette fonction permet de faire grandir le serpent après qu'il ai mangé un peu de nourriture.

▼ Indice 1

Regardez dans `constructor` si une variable peut être utile.

▼ Indice 2

La variable à modifier est `len` .

▼ Solution

```
grow() {
  let head = this.body[this.len - 1].copy();
  this.len++;
  this.body.push(head);
}
```

▼ eat()

Cette fonction permet de vérifier si le serpent a mangé, a.k.a. la tête du serpent se trouve sur de la nourriture.

▼ Indice 1

la tête et la nourriture possèdent des valeurs permettant de connaître leur position.

▼ Indice 2

La tête se trouve en `head.x` et `head.y`. à vous de deviner pour la nourriture.

▼ Solution

```
eat(food) {
  let head = this.body[this.len - 1];

  if (head.x == food.x && head.y == food.y) {
    this.grow();
    return true;
  }
  return false;
}
```

Mi casa es tu casa

Rendez-vous maintenant dans le fichier `sketch.js`. Ceci est le fichier principal du projet, celui où toutes les actions se déroulent, celui où la magie opère.

Ici, diverses fonctions vous attendent :

- `setup` : Se lance une fois en début de programme. Sert à initialiser les différentes variables.
- `foodLocation` : Donne une place aléatoire à la nourriture.
- `keyPressed` : Se lance automatiquement quand une touche du clavier est appuyée.

- `draw` : Se lance toutes les frames. Gère l'affichage et les lancements de fonctions externes.

Comme pour le serpent, vous devez compléter les lignes commentées dans le code pour pouvoir faire fonctionner votre programme. Déroulez les onglets suivant pour accéder à des indices et à la solution si besoin :

▼ `setup()`

Ici, il vous est demandé de créer un nouveau serpent et le donner une première position à la nourriture.

▼ Indice 1

Pour créer une instance d'un objet, il faut écrire `variable = new MonObjet()`.

Il semblerait que nous nous soyons occupé d'un certain objet plus tôt dans ce projet...

▼ Indice 2

Pour la nourriture, regardez les fonctions à disposition dans le fichier, vous devriez trouver votre bonheur.

▼ **Solution**

```
function setup() {  
  createCanvas(400, 400)  
  frameRate(5)  
  
  hauteur = floor(height / resolution)  
  largeur = floor(width / resolution)  
  
  snake = new Snake();  
  foodLocation();  
}
```

▼ `foodLocation()`

Maintenant, il vous faut remplacer les 0 par une valeur aléatoire de la grille.

▼ Indice 1

N'hésitez pas à aller faire un tour dans la liste des fonctions utiles du sujet.

▼ Indice 2

Les valeurs maximales se trouvent dans les variables `hauteur` et `largeur`.

▼ Solution

```
function foodLocation() {  
  let x = floor(random(largeur))  
  let y = floor(random(hauteur))  
  
  food = createVector(x, y);  
}
```

▼ keyPressed()

Cette fonction va permettre de dire au serpent de changer de direction.

▼ Indice 1

Il y a quatre directions à faire prendre à votre serpent et une est déjà codée. Essayez d'analyser comment elle est faite pour avancer.

▼ Indice 2

Directions

<u>Aa</u> direction	# x	# y
<u>Haut</u>	0	-1
<u>Bas</u>	0	1
<u>Gauche</u>	-1	0
<u>Droite</u>	1	0

▼ Solution

```
function keyPressed() {  
  if (keyCode === UP_ARROW) {  
    snake.setDir(0, -1);  
  } else if (keyCode === DOWN_ARROW) {  
    snake.setDir(0, 1);  
  } else if (keyCode === LEFT_ARROW) {  
    snake.setDir(-1, 0);  
  } else if (keyCode === RIGHT_ARROW) {  
    snake.setDir(1, 0);  
  }  
}
```

▼ draw()

Cette fonction va lancer toutes les autres. Les étapes à remplir sont déjà dans le fichier, maintenant c'est à vous de jouer !

▼ Indice 1

Une condition se fait avec un `if` et ses composants.

Par exemple : `if (condition) { action }`

▼ Indice 2

Regardez bien toutes les fonctions que vous avez programmé jusqu'à présent et essayez de trouver lesquelles correspondent le mieux.

▼ Solution

```
function draw() {
  scale(res);
  background(220);

  if (snake.eat(food))
    foodLocation();

  snake.update();
  fill(255, 0, 0);
  rect(food.x, food.y, 1, 1);
  snake.show();

  if (snake.endGame()) {
    print("ENDGAME");
    background(255, 0, 0);
    noLoop();
  }
}
```

Conclusion

"FéliSSSSSSSSSitaSSSSSSSSSSion, graSSSSSSSSSe à vous j'ai pu manger autant de cookies que je le voulais !", vous annonce Snaky, heureux. Maintenant, il va pouvoir passer une bonne journée dans sa maison.

Pour aller plus loin

Si vous avez terminé plus tôt que prévu, voici quelques idées pour aller plus loin :

- Ajouter un système de score qui s'affiche à l'écran ou dans la console.
- Changer les couleurs du serpent.

- Ajouter des graphismes plus jolis.
- Avoir un vrai écran de fin.