

Protocole d'ajout de Graphes

Mode Jeu Libre


[Retour](#)


Configuration de partie


[Règles](#)


[Commencer](#)


Choisissez un plateau


Arbre


Cycle

Grille

Grille torique

Graphes Cordaux



Configuration inconnue



Dodécabédron

Un arbre différent


[Choisissez un fichier...](#)
Importer un graphe


Paramètres supplémentaires

Nombre de nœuds :  


Arité :  


Choisissez votre type adversaire

Jouer contre un ordinateur

Jouer à 2 joueurs

Choisissez votre camp

La Chèvre

Le collecteur de choux

Choisissez la vitesse de récolte

Vitesse de récolte :

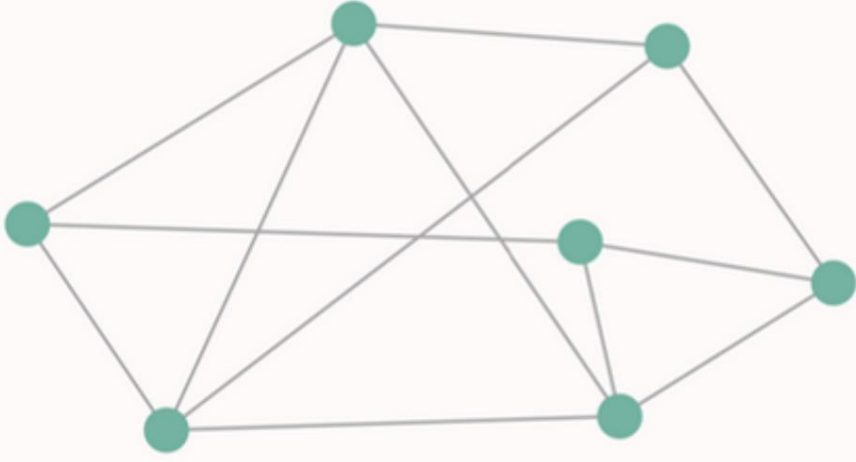
Choisissez le niveau de difficulté


[Facile](#)


Normal


Difficile


Extrême





Retour au menu

Ajouter un sommet

Ajouter une arête

Effacer un élément

Déplacer un sommet

Enregistrer

TERRA
NUMERICA

THE GAME
SURFER

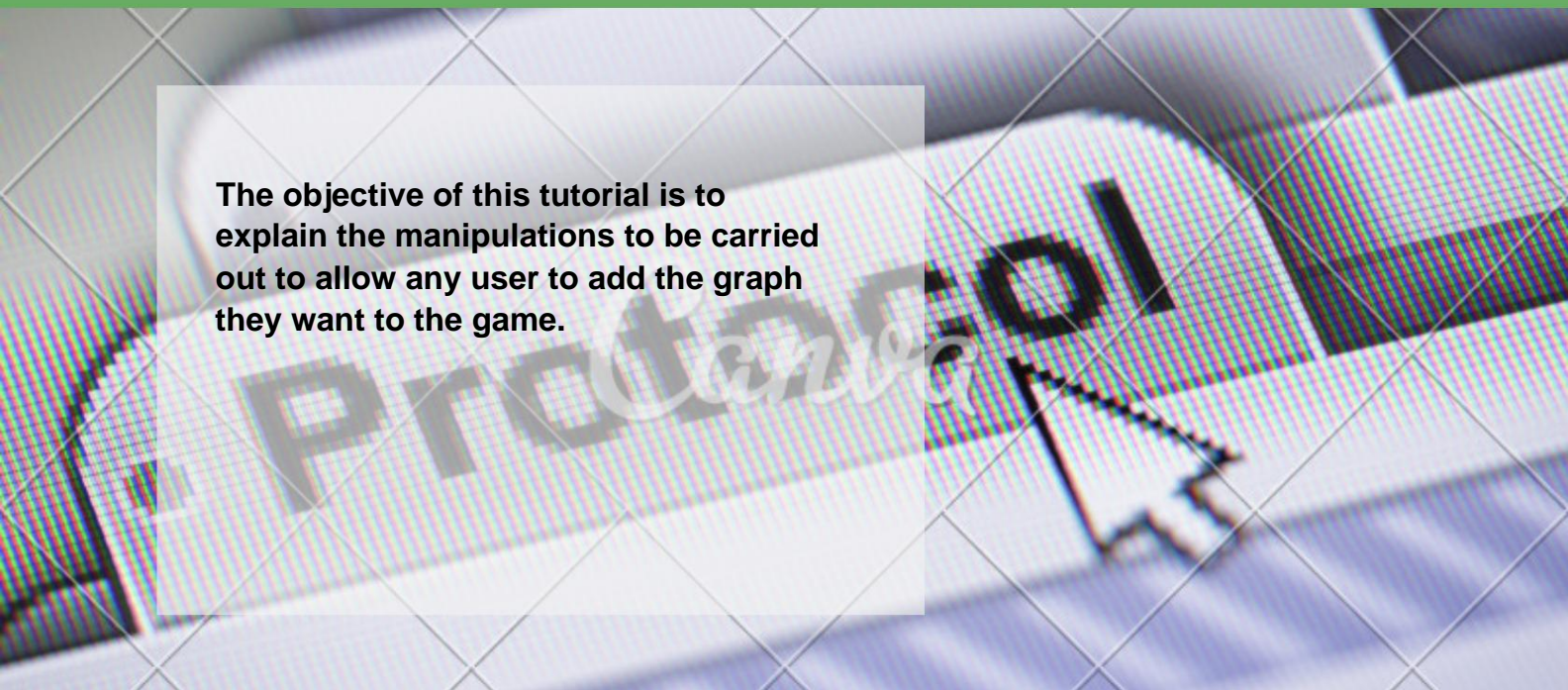
Sommaire

3 Motives

4 Quick Add Graphs

5 Adding permanent graphs

Motivations



The objective of this tutorial is to explain the manipulations to be carried out to allow any user to add the graph they want to the game.

In this Protocol, we will explain the two steps to follow to be able to insert a new graph into the game. These two ways of doing things lead to two different results, and do not involve the same investment in terms of time and memory.

The first method presented is the simplest. Everything will happen in the game application, but it will not provide a visual presentation of the graph to the user when choosing the graph.

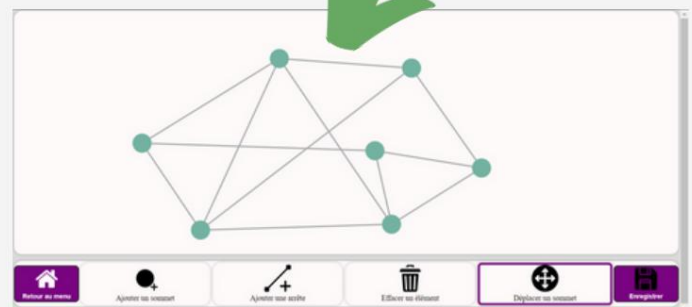
The second method is longer since you will have to go into the game code and add certain methods. However, it will allow you to display a new graph choice icon.

Méthode 1 : La méthode Rapide

First of all, the first step is to design your graph in Edit Mode, then save it to your device.



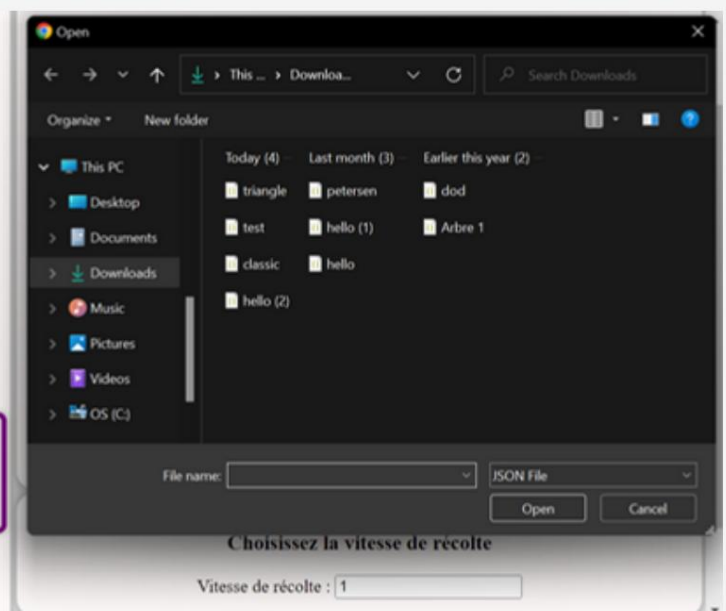
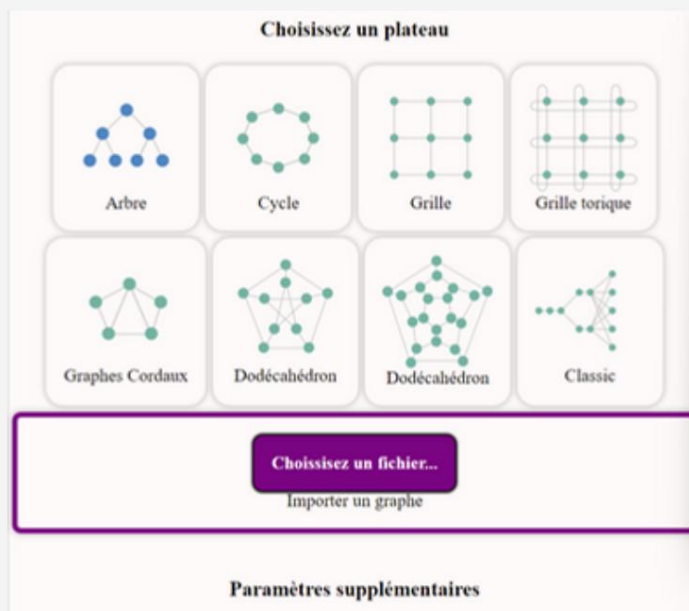
Edit Mode



Design it
Desired graph

Save it
Graph

Once saved, your graph appears under the name given in your downloads. It's a "name".json file



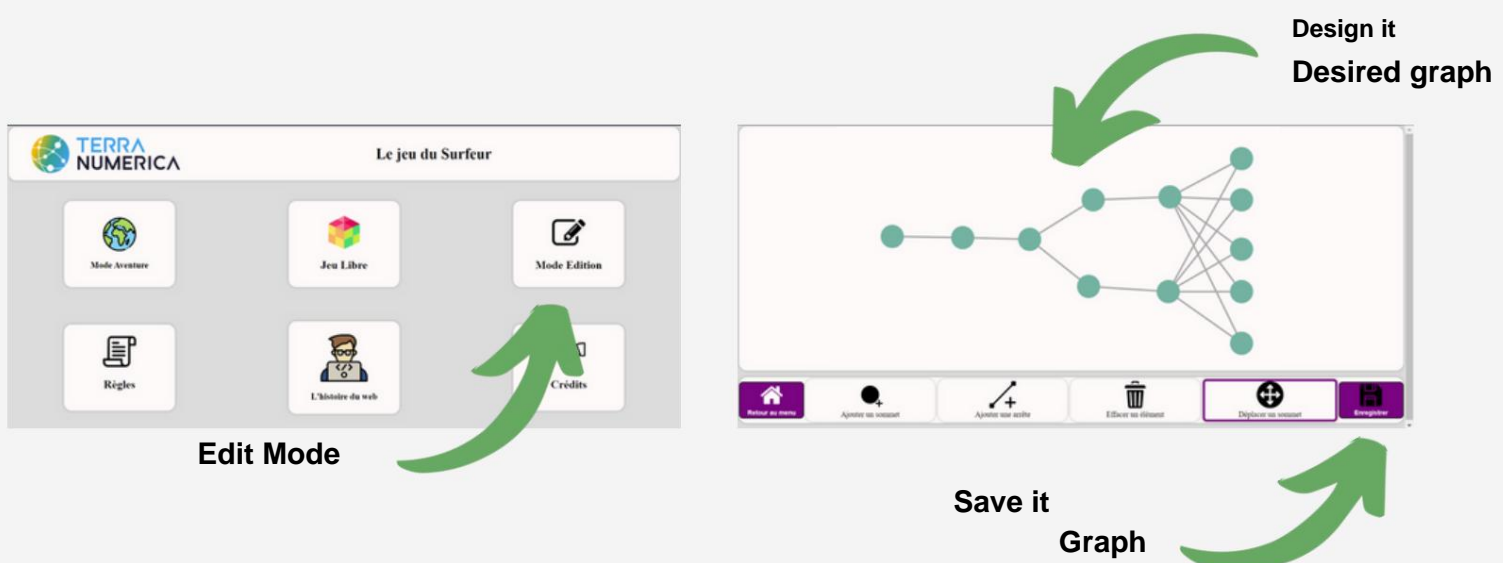
To use it, now go to Free Play Mode, and select the desired graph. To make them easier to access, you can save all the graphs designed in a directory and access them easily.

Méthode 2 :

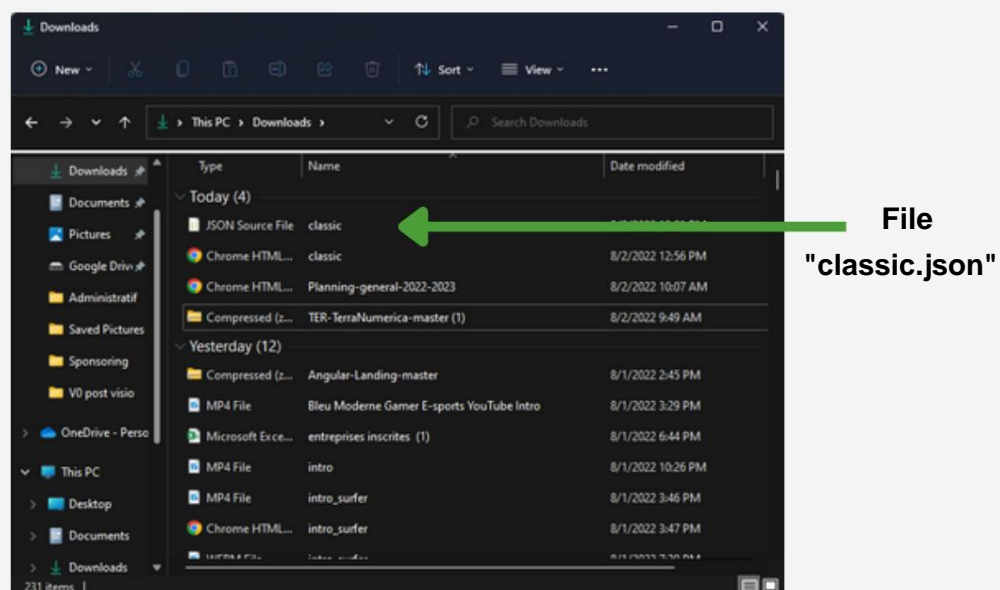
La méthode Graphique

The objective of this manipulation will be to design a display icon for the new graph created in the Free Game configuration page. In the same way as for the first method, the first step consists of designing your graph in Edit Mode, then saving it in your device.

Once the graph has been created, take a screenshot of the graph, then save it, using the **Windows Logo + Shift + S** keys (this will generate the display image)



Once saved, your graph appears under the name given in your downloads. It's a "name".json file



Méthode 2 :

La méthode Graphique

Once the graph has been created, you will need to create the display image of the graph in question. To do this, convert the .jpg image captured previously into a .svg image (opt for an online converter (<https://convertio.co/fr/jpg-svg/>))



Then, rename the downloaded image to the same name as your graph:
"name".svg

Then open your code editor (VS Code is strongly recommended since the development of the application was carried out on it).

Open the game code file, then go to **src>app>services>graph>graph.service.ts**

```

49 generateGraph(type: string, args: any[]) {
50   switch (type) {
51     case 'tree':
52       this._graph = this.generateTree(args[0], args[1]);
53       break;
54     case 'conf2':
55       this._graph = this.generateTree(args[0], 2);
56       break;
57     case 'conf':
58       this._graph = this.generateTree(args[0], args[1]);
59       break;
60     case 'grid':
61       this._graph = this.generateGrid(args[0], args[1]);
62       break;
63     case 'tore':
64       this._graph = this.generateTore(args[0], args[1]);
65       break;
66     case 'cycle':
67       this._graph = this.generateCycle(args[0]);
68       break;
69     case 'tree':
70       this._graph = this.generateTree(args[0], args[1]);
71       break;
72     case 'rope':
73       this._graph = this.oneCopsGraph(args[0]);
74       break;
75     case 'petersen':
76       this._graph = this.generatePetersen();
77       break;
78     case 'dodecahedron':
79       this._graph = this.generateDodecahedron();
80       break;
81     case 'classic':
82       this._graph = this.generateClassic();
83       break;
84     default:
85       this._graph = this.generateFromFile(type);
86       break;
87   }
88   return this._graph;
  
```

On line 49, add a 'case' to the generateGraph method, just before 'default':

box **'name':**

```

this._graph = this.generateName();
break;
  
```

Copy this code and replace "name" with the name of your Graph while retaining capital letters

Méthode 2 :

La méthode Graphique

Then go to line 420 and create a new method below generateClassic():

```
private async generateName()
{ const blob = await this.downloadAssets('name');
  const file = new File([blob], 'name.json'); /
  * console.log('FILE',file); */
  let g = this.loadGraphFromFile(file);
  this.g.param1 = -1;
  this.g.parma2 = -1; /
  * console.log('HERE'); */
  await g; }
```

```
419
420 private async generateClassic() {
421   const blob = await this.downloadAssets('classic');
422   const file = new File([blob], 'classic.json');
423   /* console.log('FILE',file); */
424   let g = this.loadGraphFromFile(file);
425   this.g.param1 = -1;
426   this.g.parma2 = -1;
427   /* console.log('HERE'); */
428   await g;
429 }
```

Then go to the file **src>app>components>configuration-menu-components.ts**

Line 16, add **'name'** to the "public configuration" list

Line 18, add a case to configuration_param_boundaries before "import":

name:

```
{ param1: { min: -1, max: -1 },
  param2: { min: -1, max: -1 } },
```

```
47  ✓ classic: {
48    param1: { min: -1, max: -1 },
49    param2: { min: -1, max: -1 }
50  },
```

Line 187, add a case to the getConfiguratonName method before "import":

box **'name':**

return **'Name'**;

```
205  ✓ case 'classic':
206    return 'Classic';
207  ✓ default:
```

Méthode 2 :

La méthode Graphique

Finally, line 242, add a condition to the if of the "selectGraphType(...)" method:

&& type !== 'name'

```

242   selectGraphType(type: string) {
243     this.selected_configuration = type;
244     if(type !== 'import' && type !== 'dodecahedron' && type !== 'petersen' && type !== 'classic') {
245       this.graphImportation = false;
246       this.graphGeneration = true;
247       this.updateGraphParams();
248     }
249     this.updateParamsName();
250   }

```

Then add the "nom.svg" and "nom.json" files to the asset directory of the game code files: **src>assets**

Finally, remember to save all the modifications you have just made to the game files, then launch the VS Code terminal (Ctrl + Shift + `), and enter the following commands:

npm install

npm run build

Once completed, the build will modify the dist folder where all the game data is located.

Then, modify the "index.html" file located in the dist folder:

Line 6:

Replace: `<base href="/">` with:

`<base href="http://www-sop.inria.fr/members/Yannis.Belkhiter/Websurfer/">`

Once this modification is made, load/replace all the files contained in dist on the server's Websurfer file.