



TIMC LAB - TEAM BCM

TUTO MODIFICATION C#

Created by : BENAÏSSA Ilian & GUCLU Sefa

Project overseen by : ANGELIQUE Stephanou

Start of project : 03/01/2022

End of project : 11/02/2022

Table des matières

1 Comment modifier le code.....	3
1.1 Modifier le script.....	3
1.2 Modifier le design.....	4
2 Comment tester le code.....	6
3 Comment compiler le code.....	7
3.1 Compilation du code.....	7
3.2 Chemin du fichier compilé.....	8

1 Comment modifier le code

Pour modifier le code, il faut ouvrir le fichier formPancreas.sln avec Visual Studio.

Nom	Modifié le	Type	Taille
.vs	27/01/2022 23:20	Dossier de fichiers	
formPancreas	09/02/2022 03:21	Dossier de fichiers	
packages	27/01/2022 23:20	Dossier de fichiers	
formPancreas.sln	27/01/2022 23:20	Visual Studio Solut...	2 Ko

1.1 Modifier le script

Une fois Visual Studio ouvert, il faut ouvrir le fichier Form1.cs pour modifier le script. Nous trouvons dans ce fichier, toutes les fonctions nécessaires au bon fonctionnement de l'application.

Exemple de la fonction qui permet de lancer la simulation via le formulaire :

valider()

```
// valider formulaire (v3)

public void valider(object sender, EventArgs e)
{
    // critères de validation de paramtres (à définir)
    if (String.IsNullOrEmpty(tb_param1.Text.Trim()) || String.IsNullOrEmpty(tb_param2.Text.Trim()) || S
    {
        MessageBox.Show("Enter all values !", "Error");
    } else if (double.Parse(tb_param1.Text)> 1) {
        MessageBox.Show("Error param 1 !", "Error");
    } else
    {
        // supprime tt les txt de la derniere simulation
        foreach (string sFile in System.IO.Directory.GetFiles(pathResults, "*.txt"))
        {
            System.IO.File.Delete(sFile);
        }

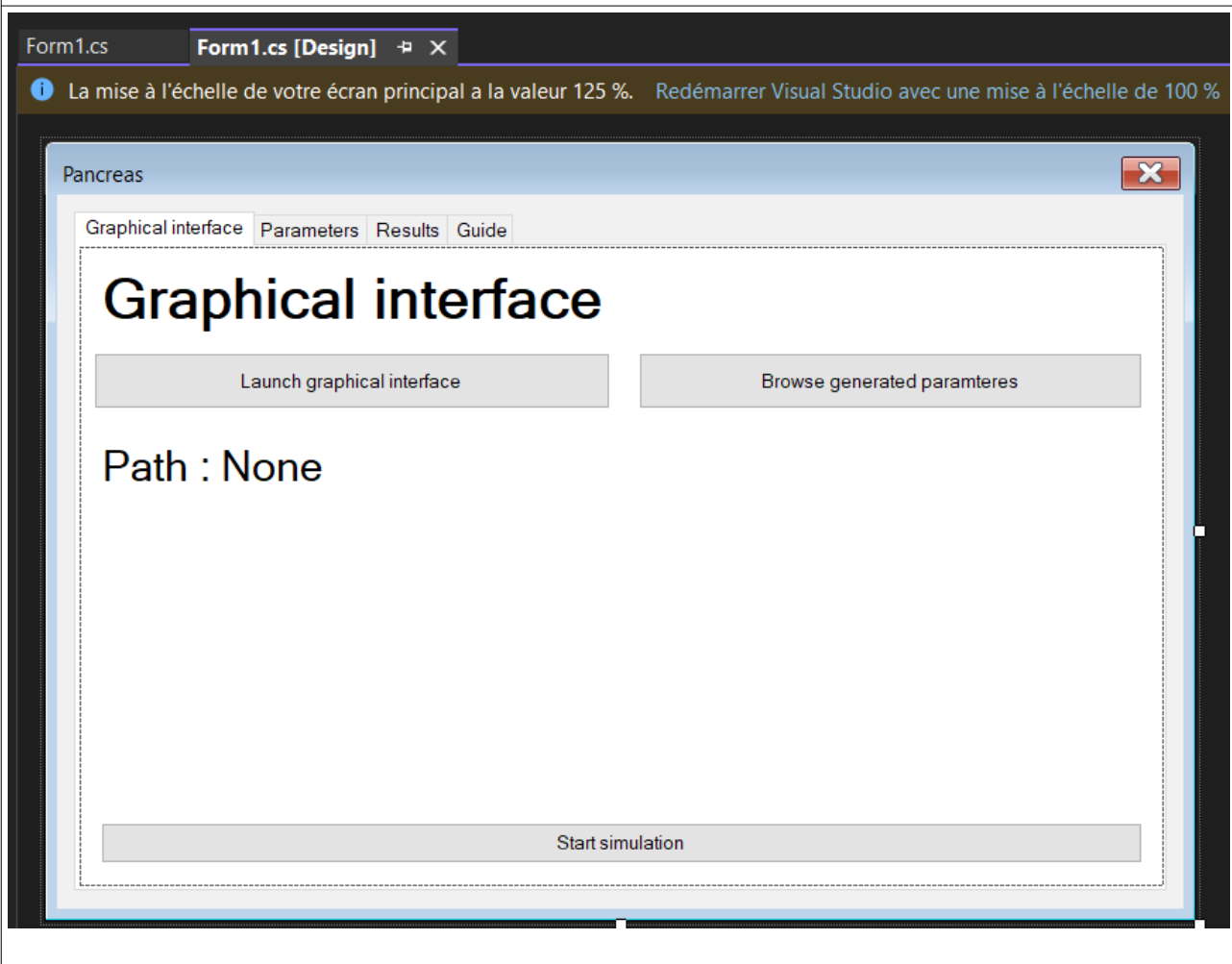
        // ecrire nouveau paramas dans un.txt
        string para = tb_param1.Text + "\n" + tb_param2.Text + "\n" + tb_param5.Text + "\n" + tb_param4
        File.WriteAllText(pathParams, para);
        File.Copy(pathParams, pathParamsResults); // copier pour para resultats

        // lancer la simulation
        Process.Start(pathExe);
    }
}
```

Par exemple, pour ajouter des critères de validation, il faudrait ajouter des conditions à la ligne 5.

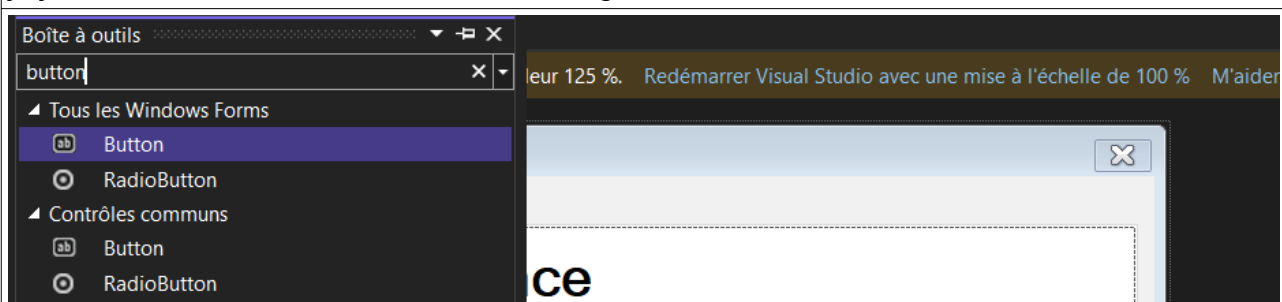
1.2 Modifier le design

Pour modifier le design de l'application, il faut ouvrir le fichier Form1.cs [Design].

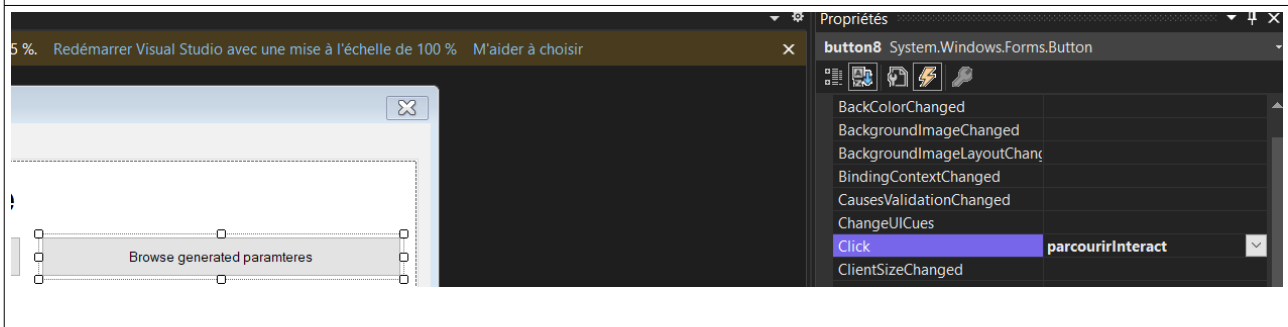


Cela ouvre une interface qui nous donne la possibilité d'ajouter des éléments (boutons etc..) et de changer leur propriétés, qui sont ensuite disponibles à être utilisé dans le script, le tout de façon graphique.

Par exemple, si je souhaite ajouter un nouveau bouton qui lance une nouvelle fonction, j'ajoute un élément « Button » à mon design.

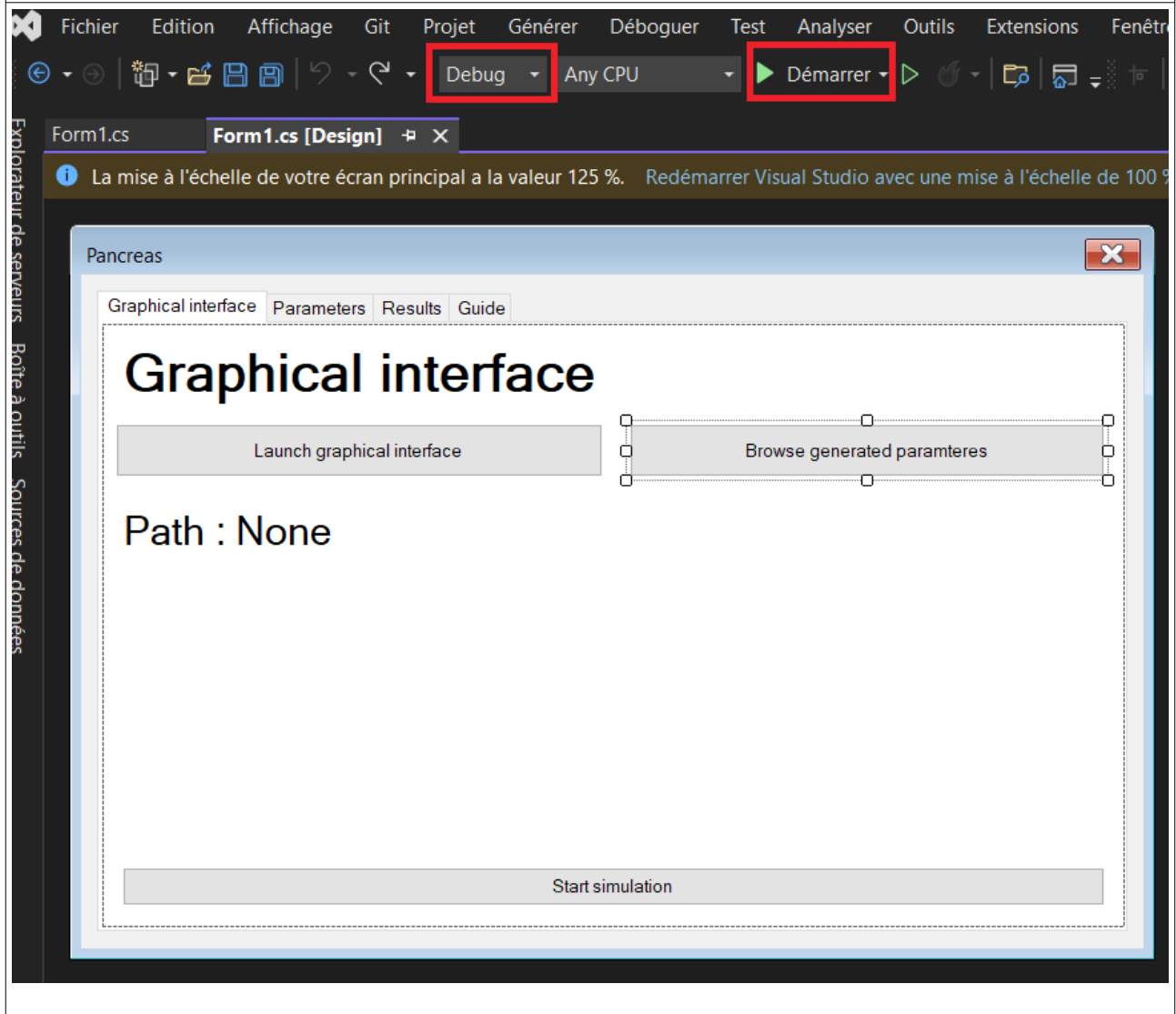


Ensuite, une fois placé, je change sa propriété « Click » en ajoutant le nom de la fonction.



2 Comment tester le code

Afin de tester le code, je clique sur la touche « F5 » ou sur le bouton « Démarrer » en haut de la page, en vérifiant bien que la case de case soit sur « Debug ».



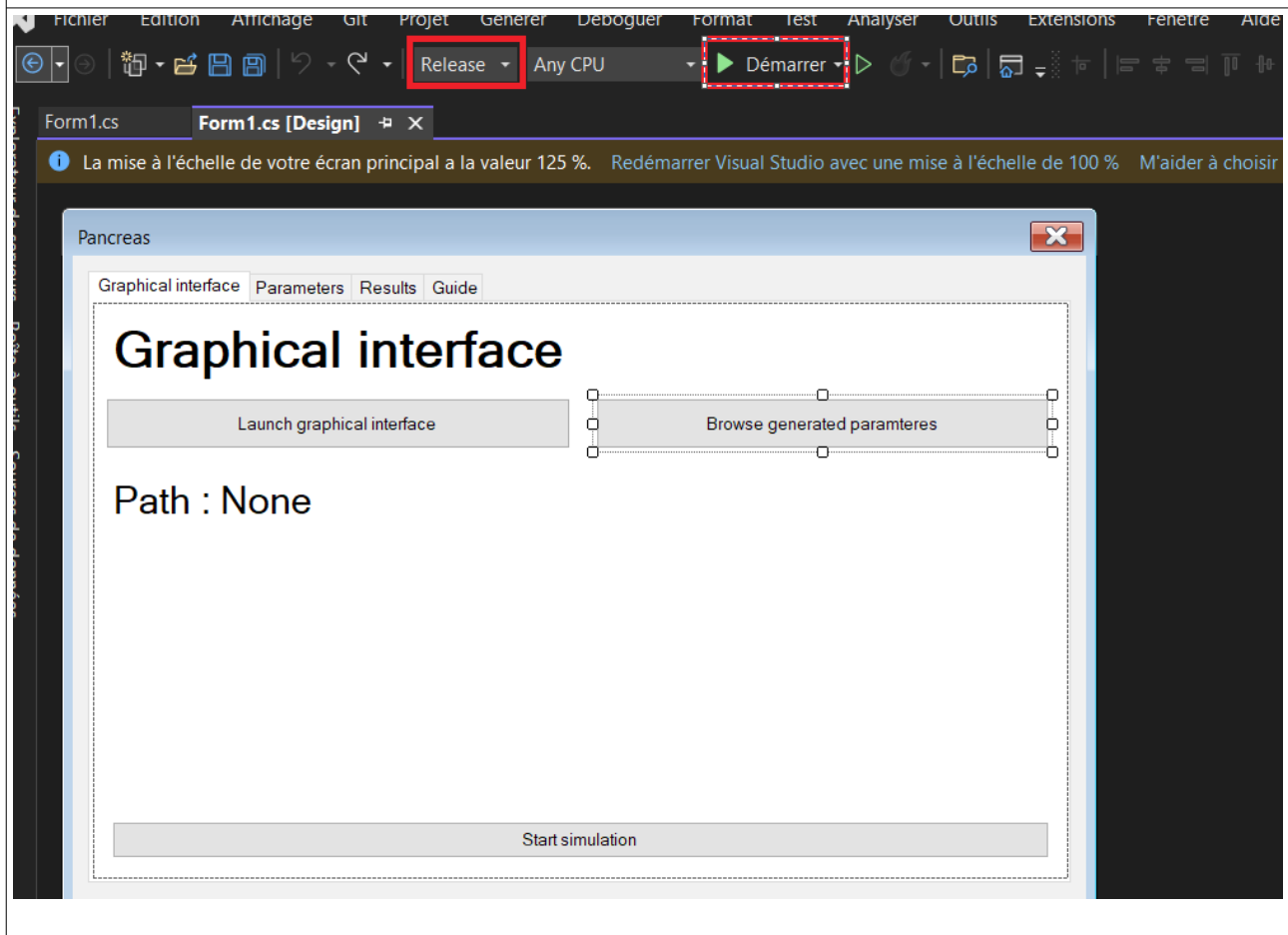
ATTENTION : il faut vérifier que les différents chemins (paths) dans le script soient valides afin de faire fonctionner le script !

3 Comment compiler le code

Une fois les modifications effectués, il faut compiler le code afin de le faire marcher.

3.1 Compilation du code

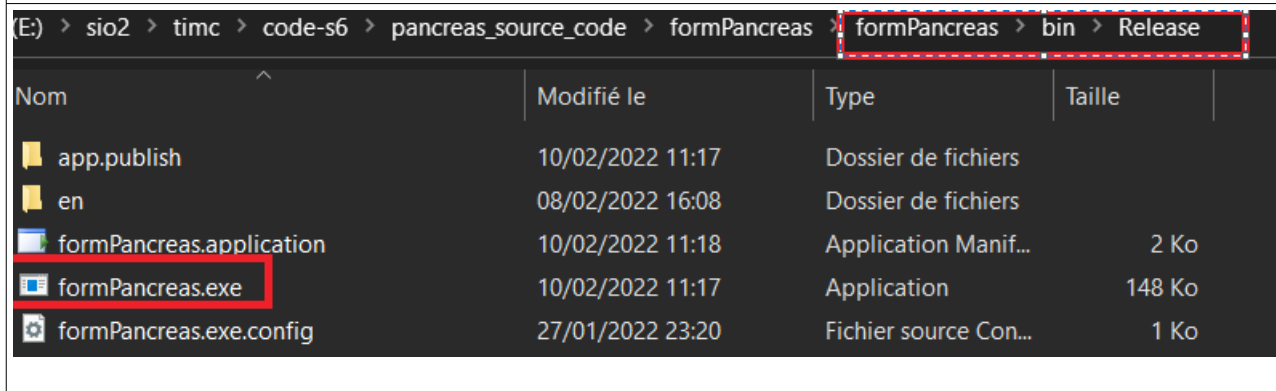
Tout comme pour le débogage, il faut cliquer sur la touche « F5 » ou sur le bouton « Démarrer » en sélectionnant cette fois « Release ».



Si aucun bug n'est détecté, la compilation devrait se compléter sans problème.

3.2 Chemin du fichier compilé

Une fois la compilation complète, le nouveau fichier formPancreas.exe se trouve dans le dossier bin/Release du même dossier.



Nom	Modifié le	Type	Taille
app.publish	10/02/2022 11:17	Dossier de fichiers	
en	08/02/2022 16:08	Dossier de fichiers	
formPancreas.application	10/02/2022 11:18	Application Manif...	2 Ko
formPancreas.exe	10/02/2022 11:17	Application	148 Ko
formPancreas.exe.config	27/01/2022 23:20	Fichier source Con...	1 Ko

Il suffit désormais de remplacer le fichier formPancreas.exe de notre dossier pancreas.

(seul ce fichier ainsi que les .dll sont importants, hors, les .dll sont déjà présents dans le dossier pancreas et n'ont donc pas besoin d'être remplacés)



TIMC LAB - TEAM BCM

TUTO MODIFICATION JS

Created by : BENAÏSSA Ilian & GUCLU Sefa

Project overseen by : ANGELIQUE Stephanou

Start of project : 03/01/2022

End of project : 11/02/2022

Table des matières

1 Fonction ajouter() :	3
2 Fonction dragMoveListener() :	5
3 Fonction modif_r() :	6
4 Fonction Lx() et Ly() :	7
5 Fonction screen :	7
6 Supprimer un cercle : Fonction supprimer()	8
7 Fonction Extraire() :	8

1 Fonction ajouter() :

Permet de créer un nouveau cercle dans la zone grise.

```
idCircle = "well-" + compteur;
ids.push(idCircle); // arrays de nos cercles
circle = `

Cette partie de la fonction crée un cercle au niveau du HTML, s'il faut modifier le HTML des cercles on passera par cette partie de la fonction.



```
volume = (Math.PI * 2*2 * document.getElementById("id_depth").value * 1000).to-
Fixed(2)
volumTotal = (2700 * (4/3) * Math.PI * (0.1**3)).toFixed(2)
rate = (volumTotal * 100 / volume).toFixed(2)
rate = `

```



La partie du code ci-dessus calcule le volume de chaque cercle (ici donc le puits), et ren-voie son % de remplissage dans le HTML.



```
ligne = `${idCircle}</td>`
 ligne += ` |
```



Cette partie de la fonction permet d'incrémenter le tableau avec les données du cercle créé. Pour rendre ces données modifiables il a suffi d'ajouter le terme «contenteditable » dans la cellule <td> HTML. Ensuite chaque frappe de clavier dans la cellule appellera une fonction différente selon la case. Ces fonctions appelées permettent changer les paramètres de chaque cercle directement au clavier.



10/02/22



3/8


```

Une fois cela fait, la fonction suivante est appelée permet d'ajouter des propriétés de « drag and drop » aux cercles :

`addInteract()`

Cette fonction est définie a la ligne 185, elle est rédigée grâce à la documentation d'interact.js . Si vous modifiez cette fonction, cela aura un impact sur le comportement de tous les cercles ainsi que la manière dont ils se déplacent dans la zone grise. Cependant vous pouvez toujours le faire grâce à la documentation disponible ici : <https://interactjs.io/docs/> .

2 Fonction dragMoveListener() :

Cette fonction permet d'enregistrer la nouvelle position x-y de chaque cercle lorsqu'on les déplace en « drag and drop ». Cette fonction récupère en continue les nouvelles positions de chaque cercle et les change en direct dans le HTML grâce au code suivant :

```
x = (parseFloat(target.getAttribute("data-x")) || 0) + event.dx,
  y = (parseFloat(target.getAttribute("data-y")) || 0) + event.dy;

target.style.webkitTransform = target.style.transform =
  "translate(" + x + "px, " + y + "px)";
// attributs
target.setAttribute("data-x", x);
target.setAttribute("data-y", y);
```

La dernière ligne permet d'appeler la fonction `resizemoica()`.

Cette fonction qui est appelée à pour objectif d'afficher correctement les données dans le tableau de données (listant chaque cercle), lorsque l'ont resize les cercles, notamment en les tirants par les extrémités.

Le code composant cette fonction est simple à comprendre, la première partie du code récupère plusieurs variables du cercle en cours de modification, la seconde partie de la fonction modifie ces variables via des calculs. La dernière partie de la fonction affiche dans le HTML les nouvelles variables modifiées.

Par ailleurs, puisqu'il est question de modification de la taille des cercles grâce à cette fonction, celle-ci se doit d'appeler une autre fonction en rapport avec le taux de remplissage de chaque puits, nommée `fillingRate()`.

Dans un premier temps, la fonction récupère la raille du rayon et la fait multiplier par 0.01 pour faire passer sa valeur de pixel a mm (selon notre propre échelle choisie).

```
// passe le rayon de pixel a mm
scale = 0.01
r = document.getElementById("r-" + interaction).innerHTML* scale
iolot = document.getElementById("n-" + interaction).innerHTML
```

Ensuite la fonction calcule le volume du puits grâce aux données récolté dans l'étape précédente:

```
//calcul du volume du puit
volume = (Math.PI * r*r * document.getElementById("id_depth").value *
1000).toFixed(2)
volumTotal = (iolot * (4/3) * Math.PI * (0.1**3)).toFixed(2)
```

Ensuite la fonction vérifie si le puits n'est pas surchargé et renvoie un code couleur en conséquence :

```
//verif si le puit nest pas surchargé
rate = (volumTotal * 100 / volume).toFixed(2)
//si surchargé > rouge sinon vert
if(rate > 100.00) {
  rate = `>${rate}%</span>`
} else {
  rate = `>${rate}%</span>`
}
document.getElementById("fr-" + interaction).innerHTML = rate
```

3 Fonction modif_r() :

Cette fonction permet de modifier au clavier directement depuis le tableau des propriétés, les propriétés de chaque cercle au cas par cas.

```
function modif_r(id) {
  newid = id.slice(2, id.length) // circle0
  newh = document.getElementById(id).innerHTML // cel

  document.getElementById(newid).style.height = newh * 2 + "px"
  document.getElementById(newid).style.width = newh * 2 + "px"

  resizemoica(newid, true)
}
```

Dans un premier temps cette fonction récupère les nouvelles données rentrées par l'utilisateur. Comme il s'agit d'un rayon ici, et que le cercle est en réalité un carré en HTML (auquel on aura arrondi les angles), on modifie alors sa longueur et sa largeur par le rayon multiplié par deux. Ils représentent le diamètre du cercle.

Pour les fonctions `modif_dx()` et `modif_dy()`, cela marche sensiblement de la même manière. Ces dernières servent à modifier au clavier la position x et y du cercle.

4 Fonction Lx() et Ly() :

```
function lx(){
  x = document.getElementById('id_x').value
  document.getElementById("zone").style.width = x+"px"
}
```

Cette fonction est appelé lorsque l'utilisateur décide de changer la taille de la zone grise depuis les paramètres globaux de la page web. Cette fonction récupère la nouvelle valeur entrée par l'utilisateur et la renvoie simplement dans le HTML afin de modifier la taille de la zone en direct.

La fonction Ly() marche exactement de la même manière.

5 Fonction screen :

La fonction screen() , utilise la librairie html2canvas, globalement cette fonction n'a pas vocation a être modifiée. Vous pouvez cependant modifier aisément le nom du .png téléchargé par l'utilisateur dans le code.

```
function screen() {
  html2canvas(document.getElementById("page_screen")).then(function (canvas)
{
  canvas.toBlob(function (blob) {
    saveAs(blob, "screenshot.png");
  });
})
}
```

6 Supprimer un cercle : Fonction supprimer()

Cette fonction permet de supprimer un cercle. Cette fonction va non seulement supprimer le cercle de la zone mais également sa ligne dans le tableau des propriétés ainsi que tout ce qui avait été lié au cercle au moment de sa création dans la fonction ajouté.

Elle n'a pas non plus vocation à être modifiée.

7 Fonction Extraire() :

Cette fonction permet de récupérer les différents paramètres rentrés par l'utilisateur ainsi que tous ceux liés aux différents cercles (puits). Une fois cela fait, ces données sont enregistrées dans un fichier nommé parameters.txt

Le nom du fichier créé est facilement modifiable ainsi que les données enregistrées dans ce fichier.



TIMC LAB - TEAM BCM

TUTO MODIFICATION PYTHON

Created by : BENAÏSSA Ilian & GUCLU Sefa

Project overseen by : ANGELIQUE Stephanou

Start of project : 03/01/2022

End of project : 11/02/2022

Table des matières

1 Comment modifier le code.....	3
2 Comment tester le code.....	4
3 Comment compiler le code.....	5

1 Comment modifier le code

Pour modifier le code, il faut ouvrir le fichier plotly-pancreas.py avec votre environnement de choix.

GB GS (E:) > sio2 > timc > code-s6 > source_code > pancreas_source_code > plotely-pancreas

Nom	Modifié le	Type	Taille
path.txt	09/02/2022 16:49	Document texte	1 Ko
plotly-pancreas.py	10/02/2022 11:20	Python file	8 Ko
seuil.txt	09/02/2022 15:19	Document texte	1 Ko

Exemple de changement : échelle absolue vers échelle relative.

```
# oxy stock valeurs pour chaque oxy.txt
# visible pour le premier, les autres false afin d'éviter bug d'affichage
# zauto = False afin de generer sa propre echelle
oxys = []
visible = True
for txt in listeTxt:
    print("Extracting "+path+txt)
    valeurs = readTxt(path, txt)
    oxys.append(go.Heatmap(z= valeurs, colorscale='rdylbu', visible = visible, zauto=False, zmin=min, zmax=max))
    visible = False
```

Il suffit de rendre zauto = True, ainsi que de supprimer zmin et zmax

2 Comment tester le code

Il suffit de lancer le script en vérifiant que les différents modules soient installés ainsi que les différents paths soient valides.

```
import plotly.graph_objs as go
import os
import re
import webbrowser
```

Pour installer plotly : pip install plotly

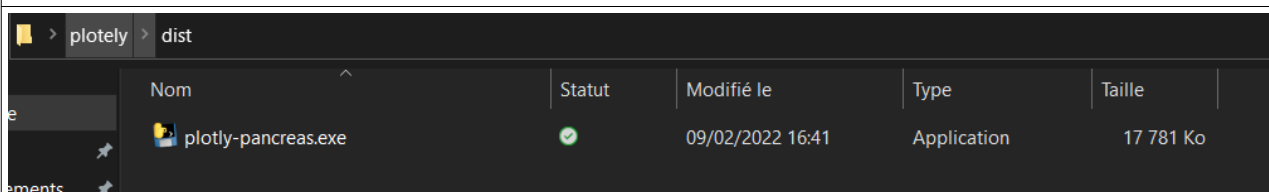
3 Comment compiler le code

Dans notre cas, nous avons utilisé pyinstaller (<https://datatofish.com/executable-pyinstaller/>) pour compiler le code, mais vous êtes libres d'utiliser la technique de votre choix.

Il suffit d'installer le module, de se placer dans le chemin de notre dossier, et de lancer la commande suivante : pyinstaller --onefile plotly-pancreas.py

```
C:\Users\sefaa\OneDrive\Bureau\plotely>pyinstaller --onefile plotly-pancreas.py
```

Une fois compilé, le fichier exe se trouve dans le dossier dist du même dossier.



Nom	Statut	Modifié le	Type	Taille
plotly-pancreas.exe	✓	09/02/2022 16:41	Application	17 781 Ko

Il suffit ensuite de remplacer l'ancien .exe par le nouveau.