SAE24 : Compte-rendu

Contents

1 Partie 1 : Installation et adaptation d'un serveur WEB 2
1.1 Introduction 2
1.2 Analyse 2
1.2.1 Contrôleur 2
1.2.2 html 4
1.2.3 Python 5
1.3 Adaptation 6
1.3.1 Le programme 6
1.3.2 Nouvelle page HTML 7
1.3.3 Les routes 7
1.4 Améliorations 9
1.4.1 Installation github 9
1.4.2 Suppression du message lors de l'initialisation de pygame 10
1.4.3 Création fonction pour créer un process python 12
1.4.4 Ajout d'une œuvre : suite carrée 13

1 Partie 1 : Installation et adaptation d'un serveur WEB

1.1 Introduction

Pour cette partie, nous avons créé une VM avec debian 11 sur les baies de virtualisation de l'IUT.

Nom VM : SA24_A1_ANDREOLI_WEB_n1

@IP : 192.168 .108.100

Utilisateur	**MDP**
Root	uMdPpS24rT22
ju	R00t24

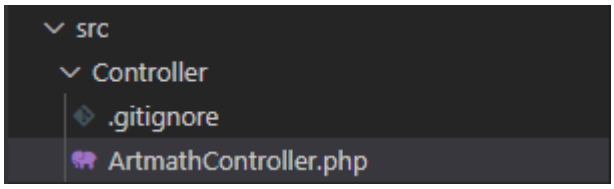
Acces site :

Utilisateur : ju

Mdp : R00t24

1.2 Analyse

1.2.1 Contrôleur



En tout premier on retrouve un contrôleur dans le projet symfony : « ArtmathController ». il permet de créer les différentes routes de notre site. On y retrouve de base les differentes routes :

- Route Racine :

```php
/**
 * @Route("/", name="racine")
 */
public function racine() : Response
{
    // Redirige vers /artmath si on va sur le site sans
    //  indiquer le nom de la route
    return $this->redirectToRoute('app_artmath');
}
```

C'est la route principale. Comme on peut le voir ci-dessus, elle redirige vers la route «artmath ». On y accède en tapant http://@IP dans un navigateur. Le serveur apache étant configuré pour écouter le port 80 (http), il redirige sur le site symfony.

- Route artmath :

```php
/**
 * @Route("/artmath", name="app_artmath")
 */
public function index(): Response
{
    return $this->render('artmath/index.html.twig', [
        'fichier' => '',
    ]);
}
```

C'est sur cette route que l'on retrouve le formulaire appelant la route qui créera l'image avec les paramètres rentrés. Cette route nous renvoi sur la page « index.html.twig »

- Route calculer :

```php
/**
 * @Route("/calculer", name="calculer")
 */
public function calculer(Request $request): Response
{
    // Récupère les paramètres issus du formulaire (on indique le champ name)
    $dimension = $request -> request -> get("dimension") ;
    // Pour les boutons : si appui contenu champ value sinon NULL
    $calculer  = $request -> request -> get("calculer");
    $imprimer  = $request -> request -> get("imprimer");


    // Oui : Appelle le script Python koch.py qui se trouve dans le répertoire /public
    $process = new Process(['python3','koch.py',$dimension]);
    $process -> run();
    // Récupère la valeur de retour renvoyé par le script python
    $fichier=$process->getOutput();

    // Retourne un message si l'éxécution c'est mal passée
    if (!$process->isSuccessful())
        return new Response ("Erreur lors de l'éxécution du script Python :<br>".$process->getErrorOutput());

    // A t'on appuyé sur calculer ?
    if ($calculer!=NULL)
        return $this->render('artmath/index.html.twig', [
            'fichier' => $fichier,
        ]);
    else {
        // On a appuyé sur imprimer
        return $this->render('artmath/imprimer.html.twig', [
            'fichier' => $fichier,
        ]);
    }
}
```

La route « calculer » appelle le programme python créant l'image à afficher.

```php
public function calculer(Request $request): Response
```

La route prend en paramètre la variable « $request », un objet de Symfony permettant de récupérer les valeurs des paramètres du formulaire.

```php
// Récupère les paramètres issus du formulaire (on indique le champ name)
$dimension = $request -> request -> get("dimension") ;
// Pour les boutons : si appui contenu champ value sinon NULL
$calculer  = $request -> request -> get("calculer");
$imprimer  = $request -> request -> get("imprimer");
```

Html:

```html
<input type="range" class="form-range" min="0" max="6" step="1" id="dimension" name="dimension">
```

Contrôleur:

```php
$dimension = $request -> request -> get("dimension") ;
```

On récupère donc les valeurs des différents champs du formulaire avec la méthode get de l'objet request. Cette méthode prend en argument le nom d'un champ du formulaire (il faut remplir l'attribut « name » sur la page html).

```php
// Oui : Appelle le script Python koch.py qui se trouve dans le répertoire /public
$process = new Process(['python3','koch.py',$dimension]);
$process -> run();
// Récupère la valeur de retour renvoyé par le script python
$fichier=$process->getOutput();
```

Pour la création de l'image avec python, on créer un nouvel objet symfony : l'objet Process. Il permet de lancer des commandes dans un sous processus sur le serveur. Lors de la création de l'objet, on doit lui passer les différents paramètres de la commande. Ici on veut lancer python donc on passe en premier paramètre « python3 » puis on met le nom du script que l'on souhaite lancer (se trouvant dans le dossier public) et enfin on met les paramètres à passer au script. Pour lancer la commande, il faut ensuite appeler la méthode « run » de l'objet Process. Enfin pour récupérer la sortie standard, on utilise la méthode « getOutput ».

1.2.2 html

- Page index.html.twig

```twig
<form method="post" action="/calculer">
    <div>
        <label for="Dimension" class="form-label">Dimension</label>
            <input type="range" class="form-range" min="0" max="6" step="1" id="dimension" name="dimension">
    </div>
    <div>
        <button type="submit" class="btn btn-primary" name="calculer" value="1">Calculer</button>
        <button type="submit" class="btn btn-danger" name="imprimer" value="1">Imprimer</button>
    </div>
</form>

<h1> Résultat : </h1>
<img src="{{fichier}}" width="100%">
<script>
```

C'est sur cette page que l'on retrouve le formulaire et l'affichage du résultat pour la première œuvre.

Le formulaire utilise la méthode post et appelle la route « /calculer ». L'attribut méthode permet de définir la méthode http (POST, GET, …) pour envoyer les données au serveur. Ici on utilise la méthode « post » pour envoyer les données du formulaire directement dans le corp de la requête.

```twig
<img src="{{fichier}}" width="100%">
```

Enfin on retrouve la balise « img » ayant pour source une variable twig nommée fichier. C'est la variable passée en paramètre lors du rendu de la page (lors du premier appel de la page, cette variable vaut une chaîne de caractère vide, puis elle aura comme valeur le nom du fichier génerer par le programme python).

1.2.3 Python

```python
import sys # Pour récupérer les paramètres passés en ligne de commande
import matplotlib.pyplot as plt
import math
```
1

```python
def motif(fig,dim,x1,y1,x2,y2):
    if dim==0:
        # Dimension = 0 -> trace une droite
        plt.plot([x1,x2],[y1,y2])
    else:
        # Sinon partage en 4 sections
        deltax=(x2-x1)/3
        deltay=(y2-y1)/3
        xa=x1+deltax
        ya=y1+deltay
        xb=xa+deltax*0.5-deltay*math.sqrt(3)/2
        yb=ya+deltax*math.sqrt(3)/2+deltay*0.5
        xc=x2-deltax
        yc=y2-deltay

        motif(fig,dim-1,x1,y1,xa,ya)
        motif(fig,dim-1,xa,ya,xb,yb)
        motif(fig,dim-1,xb,yb,xc,yc)
        motif(fig,dim-1,xc,yc,x2,y2)
```
2

```python
dimension=int(sys.argv[1]) # Premier paramètre : dimension que l'on convertit en entier
```
3

```python
# Dessine 1
fig=plt.figure()
motif(fig,dimension,0,0,1,0)
```
4

```python
# Enregistre la figure
fichier='reponse.svg'
fig.savefig(fichier)
# Ecrit le nom du fichier pour celui qui appelle ce programme
print (fichier)
```
5

1. Importation des bibliothèques python
2. Fonction récursive permettant de dessiner le motif
3. Récupération du paramètre donné avec le script dans une variable.
4. Création d'une figure pyplot et appelle de la fonction motif. C'est sur la figure que les modifications sont faites.
5. Sauvegarde de la figure pyplot et envoi le nom du fichier sur la sortie standard (stdout).

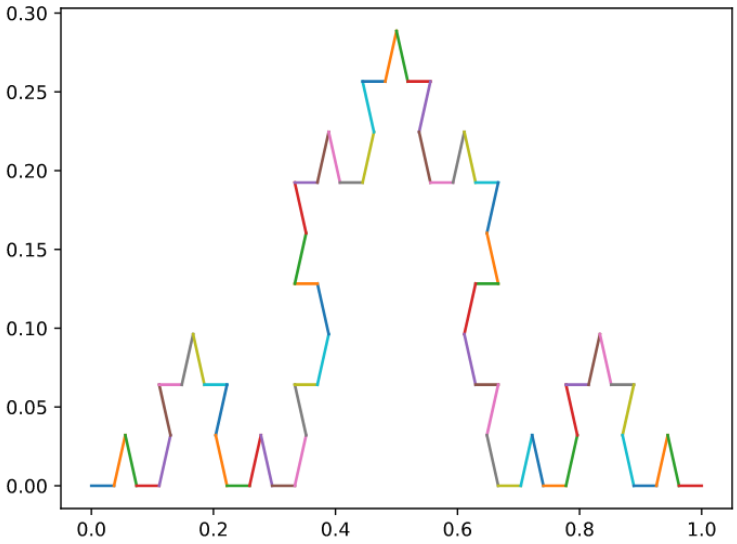
Resultat :

Paramètres :

Dimension

Calculer Imprimer

Résultat :



1.3 Adaptation

Dans cette partie nous allons rajouter une page avec un formulaire qui lancera un nouveau programme : « nees_carre.py ».

1.3.1 Le programme

```
# replace permet de transformer 5,4 en 5.4 car python attend un . pour les flottants
hasard=float(sys.argv[1].replace(',','.'))
# Amplitude des rotations
hasardangle=float(sys.argv[2].replace(',','.'))
# Nombre de lignes et de colonnes
nbcolonnes=int(sys.argv[3])
nblignes=int(sys.argv[4])

# Paramètres figés dans le programme
taille=80 # Taille d'un carré
rempli=1 # Remplissage du carré

# Taille de la zone pour dessiner
larg=taille*(nbcolonnes+2)
haut=taille*(nblignes+2)

# Initialise pygame
pygame.init()

# Crée un surface blanche où dessiner
screen = pygame.surface.Surface((larg,haut))
pygame.draw.rect(screen,(255,255,255),(0,0,larg,haut))
#screen.fill((124,125,125))

# On dessine les carrés
for i in range(nbcolonnes):
    for j in range(nblignes):
        angle=pi/4+(random.random()-0.5)*(hasardangle*j/nblignes*1.5)**3
        x0=taille+taille*(i+(random.random()-0.5)*(hasard*j/nblignes*1.5)**3)
        y0=taille+taille*(j+(random.random()-0.5)*(hasard*j/nblignes*1.5)**3)
        x1=x0+sqrt(2)/2*taille*cos(angle)
        y1=y0+sqrt(2)/2*taille*sin(angle)
        x2=x0-sqrt(2)/2*taille*sin(angle)
        y2=y0+sqrt(2)/2*taille*cos(angle)
        x3=x0-sqrt(2)/2*taille*cos(angle)
        y3=y0-sqrt(2)/2*taille*sin(angle)
        x4=x0+sqrt(2)/2*taille*sin(angle)
        y4=y0-sqrt(2)/2*taille*cos(angle)
        pygame.draw.polygon(screen,(0,0,0),((x1,y1),(x2,y2),(x3,y3),(x4,y4)),rempli)

Enregistre la figure
fichier='reponse.png'
pygame.image.save(screen,fichier)

# Ecrit le nom du fichier pour celui qui appelle ce programme
print (fichier)
```

Dans le programme on retrouve 4 paramètres d'entrées : l'amplitude du hasard, l'amplitude des rotations, nombre de lignes, et nombre de colonnes.

Comme dans le programme précèdent, le nom de l'image est envoyé sur la sortie standard.

1.3.2 Nouvelle page HTML

```
<div>
    <h1>Paramètres :</h1>
    <!-- C'est grâce à l'attribut name que l'on peut récupérer la valeur associée dans symfony -->
    <form method="post" action="/calculer_nee_carre">
        <div>
            <label for="staticAmp_hasard" class="form-label">Amplitude hasard</label>
                <input type="range" class="form-range" min="0" max="1" step="0.1" id="staticAmp_hasard" name="amp_hasard">
            <label for="staticAmp_rot" class="form-label">Amplitude rotation</label>
                <input type="range" class="form-range" min="0" max="1" step="0.001" id="staticAmp_rot" name="amp_rot">
            <label for="staticNb_col" class="form-label">Nombre colonnes</label>
                <input type="range" class="form-range" min="1" max="10" step="1" id="staticNb_col" name="nb_col">
            <label for="staticNb_lignes" class="form-label">Nombre lignes</label>
                <input type="range" class="form-range" min="1" max="10" step="1" id="staticNb_lignes" name="nb_lignes">
        </div>
        <div>
            <button type="submit" class="btn btn-primary" name="calculer" value="1">Calculer</button>
            <button type="submit" class="btn btn-danger" name="imprimer" value="1">Imprimer</button>
        </div>
    </form>

    <h1> Résultat : </h1>
    <img src="{{fichier}}" width="100%">
</div>
```

Pour intégrer le nouveau programme, on doit créer une nouvelle page HTML avec le formulaire. On retrouve les 4 paramètres que l'on transmettra au programme python.

1.3.3 Les routes

- Route nee_carre :

```
/**
 * @Route("/nee_carre", name="nee_carre")
 */
public function nee_carre(): Response
{
    return $this->render('artmath/nee_carre.html.twig', [
        'fichier' => '',
    ]);
}
```

Cette route ramène sur la page html du formulaire « nee_carre.html.twig » .

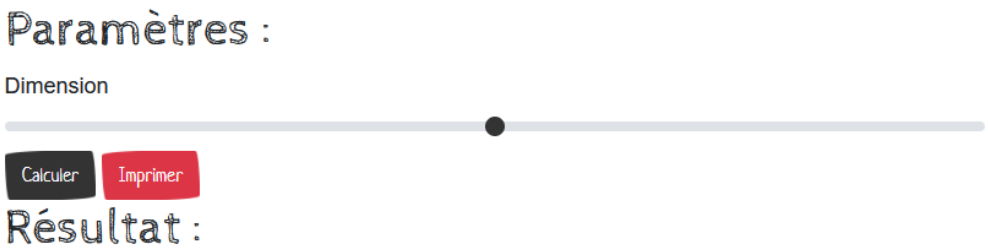
- Route calculer_nee_carre :

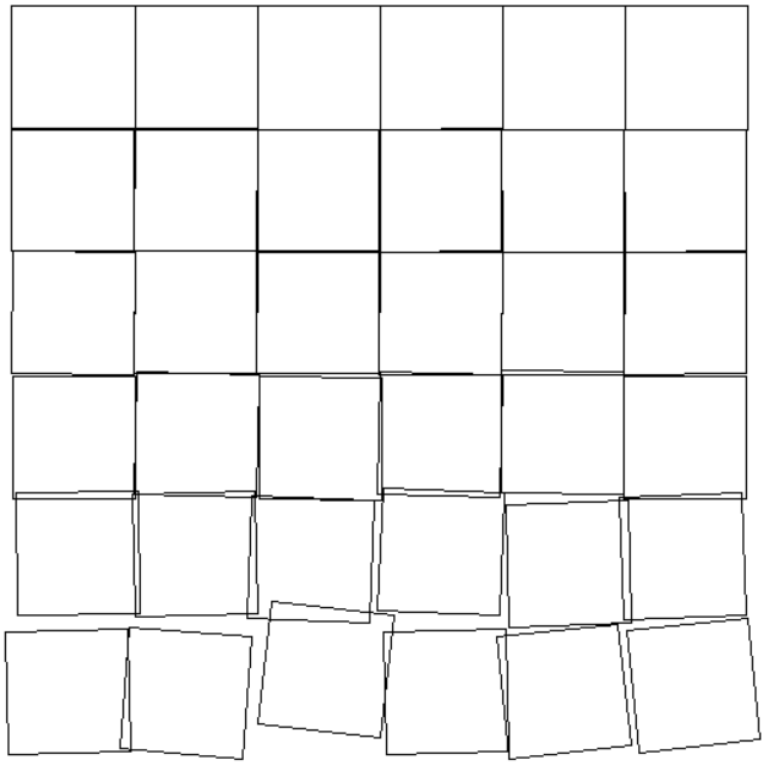
```php
/**
 * @Route("/calculer_nee_carre", name="calculer_nee_carre")
 */
public function calculer_nee_carre(Request $request): Response
{
    // Récupère les paramètres issus du formulaire (on indique le champ name)
    $amp_hasard = $request -> request -> get("amp_hasard") ;
    $amp_rot = $request -> request -> get("amp_rot") ;
    $nb_col = $request -> request -> get("nb_col") ;
    $nb_lignes = $request -> request -> get("nb_lignes") ;
    // Pour les boutons : si appui contenu champ value sinon NULL
    $calculer  = $request -> request -> get("calculer");
    $imprimer  = $request -> request -> get("imprimer");


    // Oui : Appelle le script Python koch.py qui se trouve dans le répertoire /public
    $process = new Process(['python3','nees_carre.py', $amp_hasard, $amp_rot, $nb_col, $nb_lignes]);
    $process -> run();
    // Récupère la valeur de retour renvoyé par le script python
    $fichier='reponse.png';

    // Retourne un message si l'éxécution c'est mal passée
    if (!$process->isSuccessful())
        return new Response ("Erreur lors de l'éxécution du script Python :<br>".$process->getErrorOutput());

    // A t'on appuyé sur calculer ?
    if ($calculer!=NULL)
        return $this->render('artmath/index.html.twig', [
            'fichier' => $fichier,
        ]);
    else {
        // On a appuyé sur imprimer
        return $this->render('artmath/imprimer.html.twig', [
            'fichier' => $fichier,
        ]);
    }
}
```

Cette route va appeler le programme python créant l'œuvre.

1. On récupère les nouveaux paramètres qui sont envoyés dans le corp de la requête.
2. On créer le process python avec les nouveaux paramètres
3. Cette fois la variable fichier n'est pas récupéré de la sortie standard du programme python mais est directement écrite. Cette méthode est utilisée afin de passer le problème du message d'accueil de pygame lors de l'initialisation (lors de son initialisation, pygame envoie un message sur la sortie standard)

Resultat :





1.4 Améliorations

1.4.1 Installation github

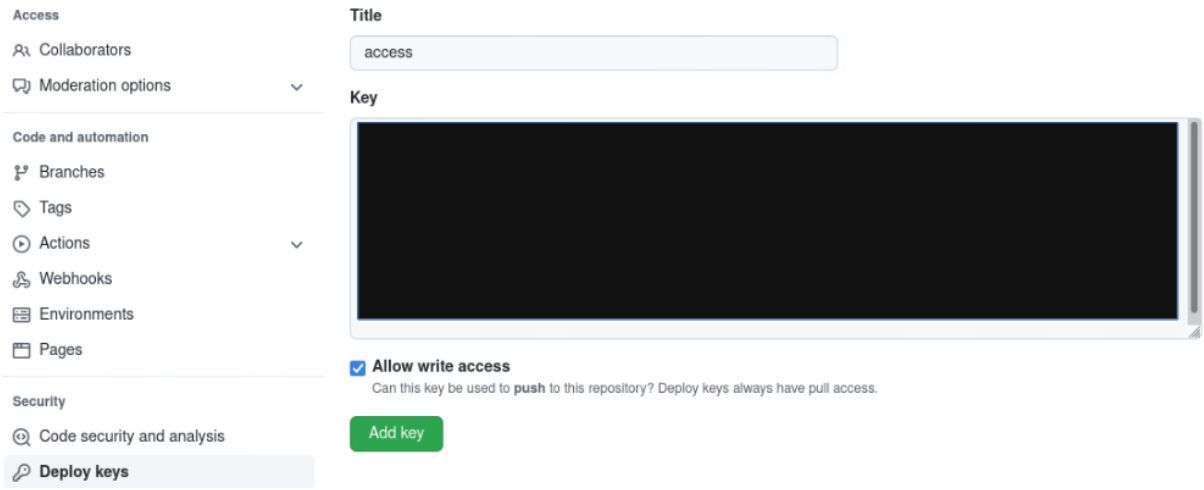
Afin de faciliter le développement du site, j'ai décider de mettre en place mon environnement git sur le serveur.

```
root@debian-ANDREOLI:/var/www/html/amath# git config --global user.name qwertyjuju
root@debian-ANDREOLI:/var/www/html/amath# git config --global user.email juju.partage@gmail.com
```

Tout d'abord, il faut créer les deux variables dans git config afin de spécifier le nom d'utilisateur (« user.name ») et l'email (« user.email »).

```
root@debian-ANDREOLI:/var/www/html/amath# ssh-keygen -C "juju.partage@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
```

Ensuite on créer une clé SSH.



On ajoute la clé SSH dans les clés de déploiement du projet.

```
 GNU nano 5.4
Host github.com
        Hostname ssh.github.com
        port 443
```

Afin que l'on puisse se connecter à github en ssh, il faut créer le fichier « config » dans le dossier « /root/.ssh » avec les lignes ci-dessus.

```
root@debian-ANDREOLI:/var/www/html/amath# ssh -T git@github.com
Hi qwertyjuju/SAE24! You've successfully authenticated, but GitHub does not provide shell access.
```

On test ensuite la configuration ssh avec la commande ci-dessus. On voit que l'authentification a bien fonctionné.

On peut maintenant utiliser les commandes git. (si le git remote n'est pas configuré, il faudra le configuré)

1.4.2 Suppression du message lors de l'initialisation de pygame

```python
import os
os.environ['PYGAME_HIDE_SUPPORT_PROMPT'] = "hide"
```

Afin de supprimer le message d'initialisation de pygame (lors de l'importation du module), il suffit de rajouter les lignes ci-dessus dans le programme python avant l'importation du module pygame. Le message d'acceuil ne sera pas envoyé sur la sortie standard et donc on pourra bien récupérer la valeur que l'on souhaite (et seulement celle-ci) dans la sortie standard.

1.4.3 Ajout Barre de navigation

J'ai choisi de rajouter une barre de navigation afin de pouvoir naviguer entre les différentes pages du site.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="container-fluid">
        <a class="navbar-brand" href="/">Menu</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarColor03" aria-controls="navbarColor03" aria-expanded="false" aria-label=
        <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarColor03">
        <ul class="navbar-nav me-auto">
            <li class="nav-item">
            <a class="nav-link" href="/koch">Koch</a>
            </li>
            <li class="nav-item">
            <a class="nav-link" href="/nee_carre">Nee carree</a>
            </li>
            <li class="nav-item">
            <a class="nav-link" href="/suite_carre">Suite carre</a>
            </li>
            <li class="nav-item">
            <a class="nav-link" href="/cr">compte rendu</a>
            </li>
        </ul>
    </div>
</nav>
```

Résultat :

1.4.4 Mémorisation formulaire

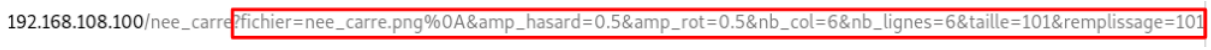
Pour la mémorisation des valeurs des formulaires plusieurs solutions sont possibles. J'ai choisi de le faire avec les fonctionnalités de twig.

1.4.4.1 Routes

- Route de calcul :

```
/**
 * @Route("/calculer_nee_carre ", name="calculer_nee_carre")
 */
public function calculer_nee_carre(Request $request): Response
{
    // Récupère les paramètres issus du formulaire (on indique le champ name)
    $amp_hasard = $request -> request -> get("amp_hasard") ;
    $amp_rot = $request -> request -> get("amp_rot") ;
    $nb_col = $request -> request -> get("nb_col") ;
    $nb_lignes = $request -> request -> get("nb_lignes") ;
    $taille = $request -> request -> get("taille");
    $remplissage = $request -> request -> get("remplissage");
    // Pour les boutons : si appui contenu champ value sinon NULL
    $calculer  = $request -> request -> get("calculer");
    $imprimer  = $request -> request -> get("imprimer");
    $out = $this->create_pyprocess("nees_carre.py", $amp_hasard, $amp_rot, $nb_col, $nb_lignes, $taille, $remplissage);
    // A t'on appuyé sur calculer ?
    if ($calculer!=NULL)
        return $this->redirectToRoute('app_nee_carre', [
            'fichier' => $out,
            'amp_hasard' => $amp_hasard,
            'amp_rot' => $amp_rot,
            'nb_col' => $nb_col,
            'nb_lignes' => $nb_lignes,
            'taille' => $taille,
            'remplissage' => $remplissage,
        ]);
    else {
        // On a appuyé sur imprimer
        return $this->render('artmath/imprimer.html.twig', [
            'fichier' => $out,
        ]);
    }
}
```

Lors du calcul des œuvres, plutôt que de juste refaire un rendu de la page du formulaire, je renvoi sur la route du formulaire en passant en paramètres les différents paramètres récupérer lors du calcul.



La redirection de route avec paramètres génère un URL avec les différents paramètres et leurs valeurs.

- Route formulaires :

```php
/**
 * @Route("/nee_carre", name="app_nee_carre")
 */
public function nee_carre(Request $request): Response
{
    $fichier = $request->query->get('fichier');
    $amp_hasard = $request -> query -> get("amp_hasard") ;
    $amp_rot = $request -> query -> get("amp_rot") ;
    $nb_col = $request -> query -> get("nb_col") ;
    $nb_lignes = $request -> query -> get("nb_lignes") ;
    $taille = $request -> query -> get("taille");
    $remplissage = $request -> query -> get("remplissage");
    return $this->render('artmath/nee_carre.html.twig', [
        'fichier' => $fichier,
        'amp_hasard' => $amp_hasard,
        'amp_rot' => $amp_rot,
        'nb_col' => $nb_col,
        'nb_lignes' => $nb_lignes,
        'taille' => $taille,
        'remplissage' => $remplissage,
    ]);
}
```

Dans les routes des formulaires, j'ai rajouté une requête récupérant les différents paramètres dans l'url (grâce à la méthode query des requêtes).

1.4.4.2 HTML

```twig
<div>
    <label for="Dimension" class="form-label">Dimension</label>
        <input type="range" class="form-range" min="0" max="6" step="1" value ="{{dimension}}" id="dimension" name="dimension">
</div>
```

Dans les formulaires, j'ai rajouté les variables twigs transférés lors du rendu de la page.

1.4.5 Création fonction pour créer un process python

```php
public function create_pyprocess(...$args){
    array_unshift($args,'python3');
    $process = new Process($args);
    $process -> run();
    $output=$process->getOutput();
    if (!$process->isSuccessful())
        throw new Exception("Erreur lors de l'éxécution du script Python :<br>".$process->getErrorOutput());
    return $output;
}
```

Afin de ne pas se répéter plusieurs fois dans le contrôleur pour la création d'un process python, j'ai choisi de créer une fonction pouvant être utilisé plusieurs fois créant un process python avec comme arguments les arguments passés à la fonction. Si une erreur est détectée, la fonction créer une erreur symfony (l'utilisateur est renvoyé sur une page de debug symfony). Si aucun problème n'est détecté, la fonction retourne la sortie standard du process.

```php
/**
 * @Route("/calculer_nee_carre ", name="calculer_nee_carre")
 */
public function calculer_nee_carre(Request $request): Response
{
    // Récupère les paramètres issus du formulaire (on indique le champ name)
    $amp_hasard = $request -> request -> get("amp_hasard") ;
    $amp_rot = $request -> request -> get("amp_rot") ;
    $nb_col = $request -> request -> get("nb_col") ;
    $nb_lignes = $request -> request -> get("nb_lignes") ;
    $taille = $request -> request -> get("taille");
    $remplissage = $request -> request -> get("remplissage");
    // Pour les boutons : si appui contenu champ value sinon NULL
    $calculer  = $request -> request -> get("calculer");
    $imprimer  = $request -> request -> get("imprimer");
    $out = $this->create_pyprocess("nees_carre.py", $amp_hasard, $amp_rot, $nb_col, $nb_lignes, $taille, $remplissage);
    // A t'on appuyé sur calculer ?
    if ($calculer!=NULL)
        return $this->redirectToRoute('app_nee_carre', [
            'fichier' => $out,
            'amp_hasard' => $amp_hasard,
            'amp_rot' => $amp_rot,
            'nb_col' => $nb_col,
            'nb_lignes' => $nb_lignes,
            'taille' => $taille,
            'remplissage' => $remplissage,
        ]);
    else {
        // On a appuyé sur imprimer
        return $this->render('artmath/imprimer.html.twig', [
            'fichier' => $out,
        ]);
    }
}
```

Dans les routes de calculs des œuvres d'art, on supprime les lignes de création de process et on les remplace par une ligne appelant la fonction nouvellement créée.

1.4.6 Ajout d'une œuvre : suite carrée

En dehors de la partie 2 de cette SAE, j'ai décidé de créer mon propre programme python afin de générer une œuvre représentant une suite de carrée.

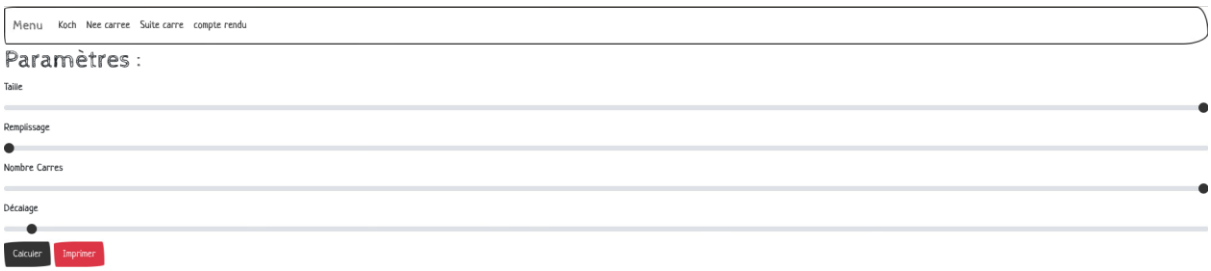
```python
import os
os.environ['PYGAME_HIDE_SUPPORT_PROMPT'] = "hide"
import sys
import random as rd
import pygame as pg
from pygame.math import Vector2 as Vec

# paramètres
taille = int(sys.argv[1])
remplissage = int(sys.argv[2])
nb_carres= int(sys.argv[3])
d = float(sys.argv[4].replace(",", "."))

pg.init()
couleurs = [(0,0,0), (0,128,0), (255, 0, 0), (255, 102, 204), (51, 51, 255), (255, 102, 0), (255, 255, 0),(153, 102, 51)]
border_x = 5
border_y = 5
surface = pg.Surface((taille+(border_x*2), taille+(border_y*2)))
surface.fill((255, 255, 255))
points=[
    Vec(border_x, border_y),
    Vec(taille+border_x, border_y),
    Vec(taille+border_x, taille+border_y),
    Vec(border_x, taille+border_y)
]
couleur = rd.choice(couleurs)
pg.draw.polygon(surface, couleur, points, remplissage)
for i in range(nb_carres):
    saved_p0 = points[0]
    points[0] = points[0].lerp(points[1], d)
    points[1] = points[1].lerp(points[2], d)
    points[2] = points[2].lerp(points[3], d)
    points[3] = points[3].lerp(saved_p0, d)
    couleur = rd.choice(couleurs)
    pg.draw.polygon(surface, couleur, points, remplissage)
fichier = "suite_carre.png"
pg.image.save(surface, fichier)
print(fichier)
```

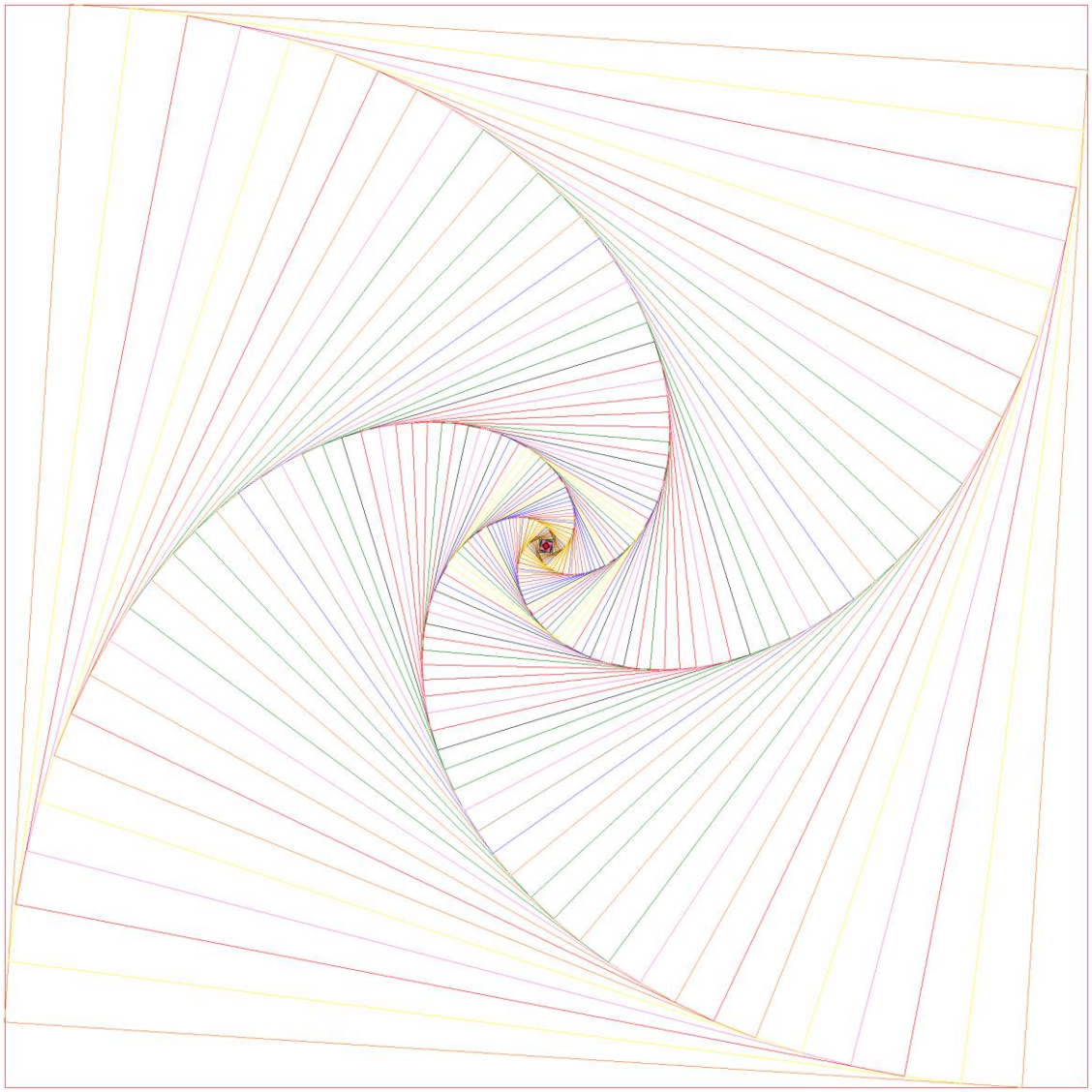
1
2
3

1. Récupération des paramètres d'entrée
2. Création de la suite carrée
3. Sauvegarde de l'image et envoi sur la sortie standard, le nom du fichier créé

Pour la partie symfony les modifications sont les mêmes que les précédentes œuvres. On modifie seulement les différents paramètres d'entrée du programme.

Résultat :





1.4.7 Ajout compte rendu

J'ai rajouté sur le site le compte rendu.

- Nouvelle route :

```php
/**
 * @Route("/cr", name="app_cr")
 */
public function cr(): Response
{
    return $this->render('artmath/cr.html.twig');
}
```

- Page html :

```
{% extends 'base.html.twig' %}

{% block title %}Art mathématique{% endblock %}

{% block body %}
<embed src="CR.pdf" width=1080 height=1920 type='application/pdf'/>
{% endblock %}
```