Anita ROCK

M1 Management de l'Innovation -Anita ROCK- anita.rock@etu.sorbonne-universite.fr

Contexte : Ces exercices ont été réalisés dans le cadre de l'UE UM4MN011 - Processus d'innovation.

Dates : Du 16 décembre 2025 au 9 février 2026.

Machines et Matériaux : Toutes les impressions ont été effectuées sur les imprimantes 3D Prusa MK4S avec du filament PLA (Rouge,Jaune, gris).

16/12/2025: Exercice 1 : Découverte de l’impression 3D à partir d’un modèle existant – Figurine de Renne

- **Consigne:** Choisir un objet dans la galerie d’objets PrusaSlicer et imprimer en 3D un objet

Objectifs :

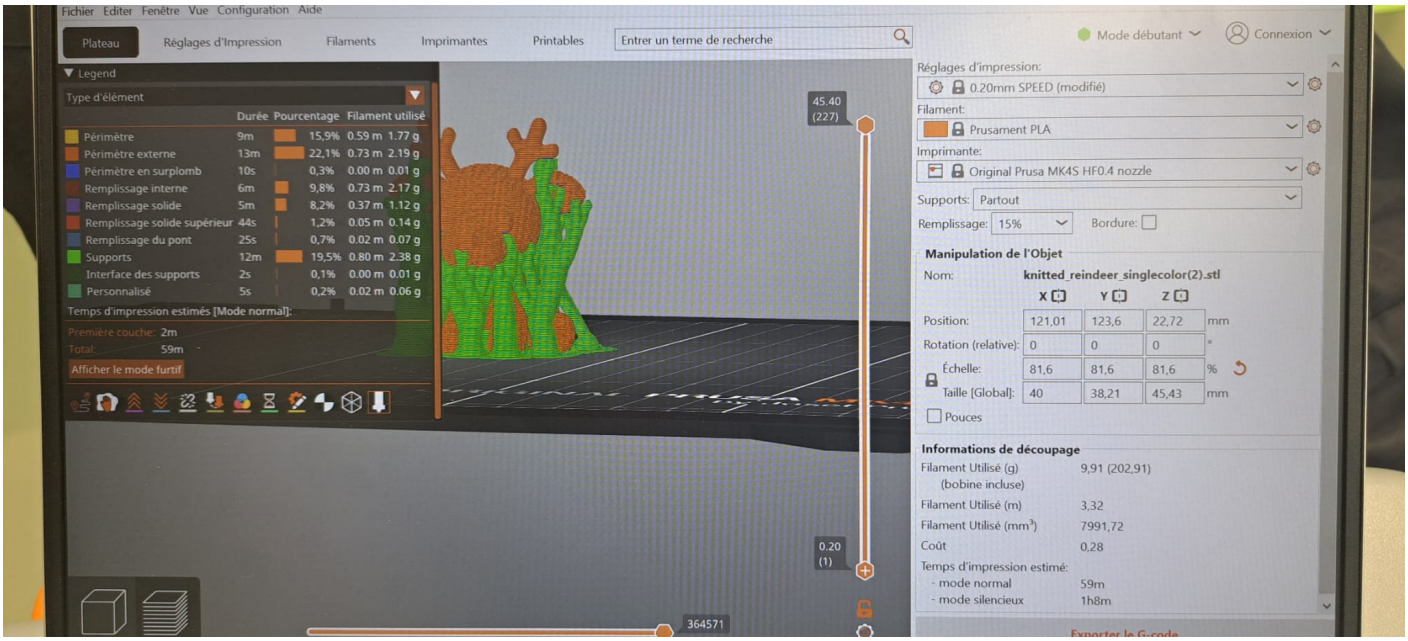
- Découvrir l’impression 3D à partir d’un modèle existant.
- Comprendre le rôle du slicer (PrusaSlicer) et l'importance des supports.
- Maîtriser l'importation de fichiers (STL/OBJ).

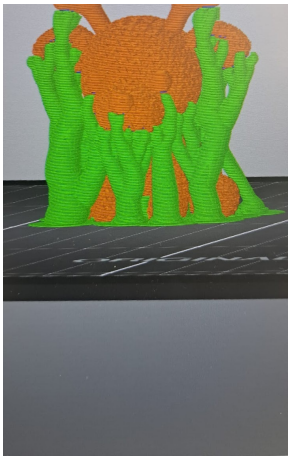
Présentation du projet : Pour ce premier exercice, j’ai choisi d’imprimer une figurine de renne issue de la bibliothèque de modèles. Ce choix permet de tester la capacité de l'imprimante à gérer des formes organiques et des parties en porte-à-faux (comme les bois du renne).

Processus d’impression : Le modèle a été importé dans PrusaSlicer via le menu Fichier > Importer STL ou bien en faisant un simple glisser-déposer du fichier sur la fenêtre du logiciel

. Pour assurer la réussite de l'impression, j'ai dû configurer les paramètres suivants :

- Supports : Ajout de supports pour soutenir les bois et le corps du renne pendant l'impression.
- Remplissage : Standard (15%).
- Matériau : PLA.





Analyse : L'ajout des supports a été l'étape cruciale. Sans eux, les parties horizontales auraient "imprimé dans le vide", causant un échec total.

13/01/2026: Exercice 2 : Apprentissage de la modélisation par le code – Pikachu

- **Consigne:** Découverte d'OpenSCAD et Créer un objet afin de se familiariser avec le code OpenScad.

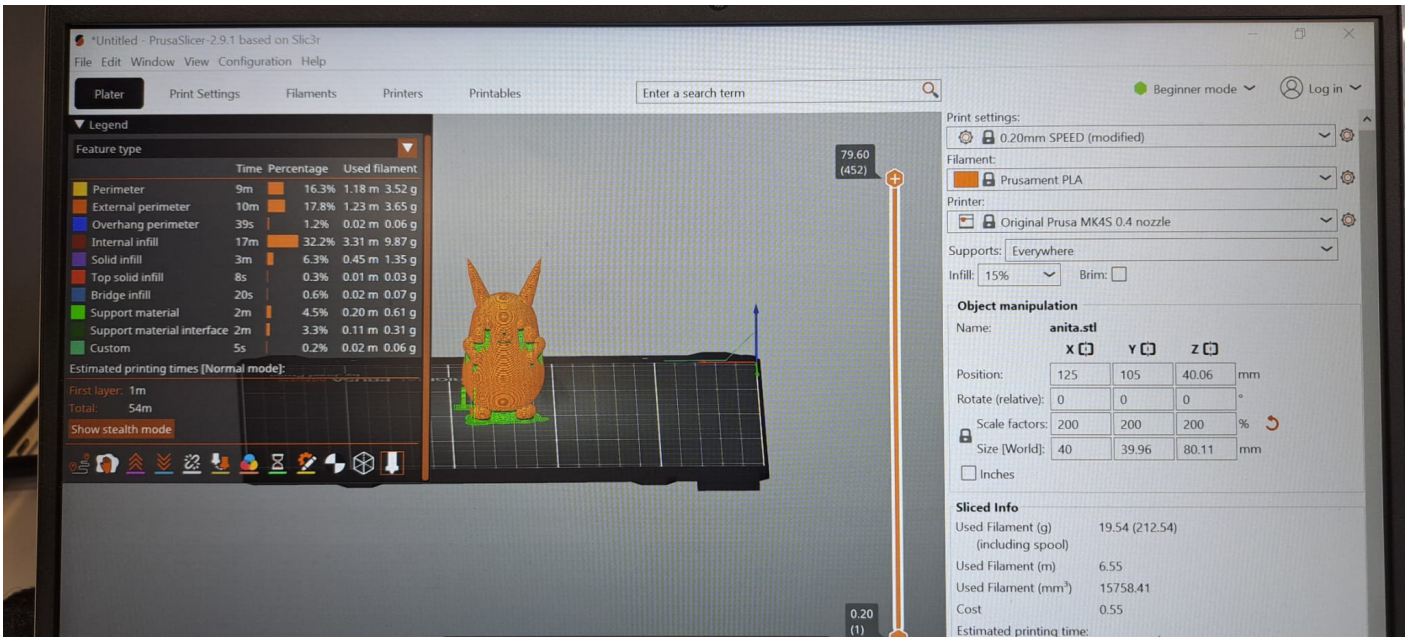
Objectifs :

- Utiliser des fonctions de codage pour modéliser un objet.
- Comprendre le processus itératif (tester, analyser, corriger).
- Gérer les contraintes de gravité et d'esthétique.

Présentation du projet : Mon objectif était de coder un Pikachu. Ce projet a nécessité plusieurs versions pour arriver à un résultat stable et fidèle au personnage.

Essai 1 – Le prototype (Fil rouge)

- Le premier essai a servi de "crash-test" pour valider la structure globale du code. Cependant, plusieurs défauts majeurs ont été identifiés, tant sur le plan technique que visuel :
  - **Contraintes de temps et de pratique (Colorimétrie) :** Le Pikachu a été imprimé intégralement en **fil rouge**. Ce choix n'était pas intentionnel : par manque de pratique sur la machine et en raison du temps limité pour cette session, je n'ai pas pu procéder au changement de filament pour obtenir le jaune emblématique. Cette première itération a donc servi à valider les volumes avant de passer à la couleur finale.
  - **Problème de stabilité et de gravité :** Le modèle était trop imposant. Cette masse excessive, couplée à un centre de gravité mal calculé dans le code, rendait la figurine instable : le Pikachu ne tenait pas debout et basculait vers l'arrière.
  - **Erreurs de modélisation (Code) :**
    - **La queue :** Mal positionnée sur l'axe de coordonnées, elle ne s'intégrait pas naturellement au corps.
    - **Le visage :** Les joues étaient mal proportionnées et la bouche était totalement difforme, résultant d'une mauvaise gestion des fonctions géométriques dans le code. De plus, un bug a généré un défaut structurel visible juste au-dessus de l'œil droit.
  - **Finition et rendu :** L'aspect général était très rugueux. Les couches étaient trop marquées et on observait un surplus de "fils" de plastique, rendant la surface peu esthétique et s'éloignant de l'aspect lisse recherché pour un personnage de ce type.



Essai 2 – Version finale optimisée (Fil jaune)

J'ai retravaillé le code pour corriger chaque point :

- **Maîtrise de la machine :** Avec plus d'assurance et une meilleure gestion du temps, j'ai pu changer le filament pour du **jaune**, rendant le modèle immédiatement plus fidèle et reconnaissable.
- **Optimisation du code et des paramètres:**
  - J'ai réduit l'échelle globale du modèle pour alléger la structure garantissant une parfaite stabilité.
  - **Correction faciale :** J'ai redessiné la bouche pour lui donner une forme cohérente, repositionné la queue et ajusté le diamètre des joues pour un rendu plus harmonieux. Le code a été nettoyé pour supprimer l'artefact au-dessus de l'œil droit.

Voici le code final amélioré et le résultat final:

```
//  Paramètres de qualité
$fn = 150;

// Couleurs
jaune = [1, 0.9, 0];
noir = [0, 0, 0];
rouge = [1, 0.2, 0.2];

module corps() {
    color(jaune)
    hull() {
        translate([0, 0, 0]) sphere(r=10);        // Bas du corps
        translate([0, 0, 12]) sphere(r=8);       // Tête
    }
}

module oreille_gauche () {

        color(jaune)
        translate([5, 0, 18])
        rotate([0, 20, 0])
        hull() {
            sphere(r=2);
            translate([0, 0, 8]) sphere(r=1);
        }{
        // Pointes noires
        color(noir)
        translate([7.7, 0, 25.5])
        rotate([0, 20, 0])
        cylinder(h=4, r1=1, r2=0);
    }}
module oreille_droite () {

        color(jaune)
        translate([-5, 0, 18])
        rotate([0, -20, 0])
        hull() {
            sphere(r=2);
            translate([0, 0, 8]) sphere(r=1);
        }{
        // Pointes noires
        color(noir)
        translate([-7.7, 0, 25.5])
        rotate([0, -20, 0])
        cylinder(h=4, r1=1, r2=0);
    }}
module joues_et_yeux_nez() {
    // Joues rouges
    for(i = [-1, 1]) {
        color(rouge)
        translate([i*6, 4, 11])
        rotate([90, 0, 0])
        sphere(r=2);

        // Yeux
        color(noir)
        translate([i*4, 7.5, 14])
        sphere(r=1.2);

        //Nez
        color(noir)
        translate([0,8.5,12])
        scale([1.5,1,1])
        sphere(r=0.4);
    }
}


module bras_gauche() {
        color(jaune)
        translate([8, 4, 8])
        rotate([-150, 40, -15])
        hull() {
            sphere(r=2);
            translate([0, 0, 4]) sphere(r=1.5);
        }
    }
module bras_droit() {
        color(jaune)
        translate([-8, 4, 8])
        rotate([-150, 40, -85])
        hull() {
            sphere(r=2);
            translate([0, 0, 4]) sphere(r=1.5);
        }
    }

module queue() {
    color(jaune)
    translate([-10, -9, 7])
    rotate([0, 90, -90])
    linear_extrude(height = 2)
    polygon(points=[[0,-2], [4,2], [2,2], [6,5], [4,5], [10,11], [10,12]
     [-2,6], [2,6], [-6,2]]);
}
module bouche(){
    color(noir)
    difference(){
 for(i = [-0.9, 0.9]) {
    translate([i*1,8.5,11])  rotate([90,0,0]) cylinder(h = 1, r = 1,
    center = true);


 }
 for(i = [-0.9, 0.9]) {
    translate([i*1,8.5,11])  rotate([90,0,0])  cylinder(h=1, r=0.75,
    center = true);


 }}}

// Assemblage
union() {
    corps();
    oreille_gauche();
    oreille_droite();
    joues_et_yeux_nez();
    bras_gauche();
    bras_droit();
    queue();


    // Pieds
    for(i = [-1, 1]) {
        color(jaune)
        translate([i*5, 2, -9])
        scale([1, 1.5, 0.6])
        sphere(r=3);
    }
}
```

Analyse : L'impression est nettement plus propre, les parois sont plus lisses et les détails sont plus fins. Le processus itératif a permis de passer d'un prototype monochrome à une figurine stable, propre et enfin fidèle à Pikachu.

Exercice 3 : Intégration d'un aimant– Magnet Marmotte

Objectifs :

- Modéliser un objet avec un logement interne.
- Utiliser la fonction de pause pour insérer un objet (aimant) pendant l'impression.

Présentation du projet : Inspirée par les marmottes découvertes lors de notre voyage au ski en Master, j'ai décidé de créer un magnet en forme de marmotte. Le défi technique était de cacher l'aimant à l'intérieur de la structure.

Processus de fabrication :

1. Conception : Création d'une cavité au centre de la structure de la marmotte pour accueillir l'aimant.
2. Mesures : Prise de mesure précise de l'aimant pour que le logement ne soit ni trop grand (jeu), ni trop petit (insertion impossible).
3. Pause à la hauteur : Programmation d'un arrêt de l'imprimante juste avant que la "toiture" de l'objet ne soit imprimée.
4. Insertion : Placement manuel de l'aimant dans la pièce, puis reprise de l'impression pour le sceller définitivement.

Voici le code et le résultat final:

```scad
// --- PARAMÈTRES AIMANT ---
aimant_diametre = 20;
aimant_epaisseur = 6;
marge = 0.2;

// --- AUTRES PARAMÈTRES ---
PI_VAL = PI;
DETAIL = 40;
CORPS_COULEUR = [0.6, 0.4, 0.2];
BONNET_COULEUR = "red";

// --- MODULES ---
module dent() {
    color("white") cube([1, 0.5, 2], center = true);
}

module patte(cote = 1) {
    color([0.3, 0.2, 0.1])
    translate([cote * 6, 0, -8])
    rotate([0, 90, 0])
    cylinder(h = 4, r = 2, center = true, $fn = DETAIL);
}

// Nouveau Module : Bonnet (Utilise Cylindre conique et Sphère)
module bonnet() {
    color(BONNET_COULEUR) {
        // Base du bonnet (Cône : r1 est la base, r2 est le sommet)
        translate([0, 8, 16])
            cylinder(h = 6, r1 = 5, r2 = 0.5, $fn = DETAIL);

        // Pompon
        translate([0, 8, 22])
            sphere(d = 2.5, $fn = 15);
    }
}

// --- CONSTRUCTION FINALE ---
union() {
    difference() {
        union() {
            // Le corps
            color(CORPS_COULEUR)
            scale([1, 0.8, 1.2])
            sphere(d = 20, $fn = DETAIL);

            // La tête
            translate([0, 8, 12]) {
                difference() {
                    color(CORPS_COULEUR) sphere(d = 12, $fn = DETAIL);
                    for (i = [-1, 1]) {
                        translate([i * 3, 4, 2]) sphere(d = 1.5, $fn = 10);
                    }
                }
                // Yeux
                for (side = [-1, 1]) {
                    translate([side * 3, 4.5, 2])
                    color(side < 0 ? "black" : [0.1, 0.1, 0.1])
                    sphere(d = 1.2, $fn = 10);
                }
                // Museau
                translate([0, 5, -1]) {
                    color([0.2, 0.1, 0]) sphere(d = 3, $fn = 10);
                    for (pos_x = [-0.6, 0.6]) {
                        translate([pos_x, 0.5, -2]) dent();
                    }
                }
            }

            // Ajout du Bonnet si la taille de la tête est suffisante
            if (DETAIL > 10 && CORPS_COULEUR[0] > 0) {
                bonnet();
            }
        }

        // --- SOUSTRACTION DE L'AIMANT ---
        translate([0, 0, -11.5])
        color("silver")
        cylinder(h = aimant_epaisseur + marge, d = aimant_diametre + marge
    , center = true, $fn = 60);
    }

    // Les pattes
    for (s = [-1, 1]) patte(s);
}

echo("Rappel : Insérez l'aimant de 20mm avant que le dos ne soit fermé !");
```

Conclusion : Cet exercice valide la capacité à mixer impression 3D et composants externes. La marmotte est parfaitement fonctionnelle et l'aimant est totalement invisible, intégré dans la masse.

Conclusion Générale : Ces trois exercices m'ont permis de passer d'une simple utilisation de modèles existants à une compréhension avancée de la modélisation (par code) et des techniques de fabrication spécifiques (supports, pause d'impression, intégration d'objets). La documentation montre que chaque erreur rencontrée (stabilité du Pikachu, mauvais placement de l'aimant) a été une étape clé du processus d'apprentissage.

Exercice 4 : Initiation à Arduino – Microcontrôleurs et Capteurs - Travail en DUO avec Savannah KADISSY

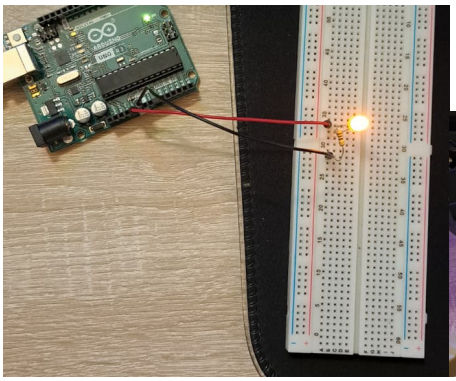
Objectifs :

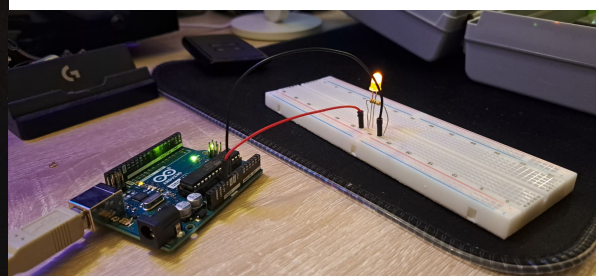
- Se familiariser avec l'environnement Arduino.
- Comprendre le passage d'un signal électrique à une ligne de code.
- Apprendre à utiliser un capteur analogique pour piloter des sorties numériques.

1. Circuit avec deux LED décorrélées : L'objectif est de faire clignoter deux LEDs à des fréquences différentes de manière indépendante.

Premier pas : Allumage simple d'une LED

Pour débuter, l'objectif était de valider la continuité du circuit électrique et la communication entre l'ordinateur et la carte Arduino.





Introduction au facteur temps : Le clignotement

Une fois la LED allumée, nous avons introduit la notion de délai avec la fonction `delay()`. L'idée est de créer un cycle infini : envoyer du courant (`HIGH`), attendre, couper le courant (`LOW`), attendre.

Voici le code et la vidéo:

```
void setup() {

pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
digitalWrite(LED_BUILTIN,HIGH);
delay(1000);
digitalWrite(LED_BUILTIN,LOW);
delay(1000);
}

```

https://www.youtube.com/embed/CXH-BrMvIAw

Extension du circuit : Deux LEDs simultanées

Nous avons complexifié le montage en ajoutant une seconde LED sur une autre broche numérique. Dans ce premier test à deux composants, le code ordonne aux deux LEDs de s'allumer et de s'éteindre exactement en même temps.

Défi : Il a fallu doubler les branchements sur la plaque d'essai (breadboard) tout en s'assurant que chaque LED possède sa propre résistance pour éviter de les griller.

```
void setup() {
pinMode(13,OUTPUT);
pinMode(7,OUTPUT);
}

void loop() {
digitalWrite(13,HIGH);
digitalWrite(7,LOW);
delay(100);

digitalWrite(13,LOW);
digitalWrite(7,HIGH);
delay(100);
}

```

https://www.youtube.com/embed/4MQC3uqGdmY

Le clignotement alterné

En adaptant le code précédent, nous avons cherché à créer un effet de va-et-vient. Pour cela, nous avons croisé les états : lorsque la LED 1 est sur `HIGH`, la LED 2 doit être sur `LOW`, et inversement.

Résultat : Ce montage permet de comprendre comment manipuler plusieurs sorties de manière logique pour créer une séquence visuelle organisée. C'est la base de la signalétique.

```
void setup() {
pinMode(13,OUTPUT);
pinMode(7,OUTPUT);
}

void loop() {
digitalWrite(13,HIGH);
digitalWrite(7,LOW);
delay(2000);

digitalWrite(13,LOW);
digitalWrite(7,HIGH);
delay(1000);
}

```

https://www.youtube.com/embed/jrg1Jr7vTp0

2. Circuit avec actionneur : L'automatisme par photo-résistance (LDR)

Pour ce dernier montage, nous avons quitté le domaine du pur clignotement pour créer un circuit qui réagit à son environnement. Nous avons intégré une **photo-résistance (LDR)**, un capteur dont la résistance varie selon la lumière reçue.

Le concept : Créer un éclairage automatique. L'Arduino lit une valeur analogique . Nous avons défini un seuil critique : si la luminosité passe en dessous de ce seuil (obscurité), la carte envoie l'ordre d'allumer la LED.

Analyse : Ce projet simule parfaitement le fonctionnement des lampadaires de rue ou des écrans de smartphone qui s'adaptent à la lumière ambiante.

```
int photo = A0;
int led = 8;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(photo, INPUT);
  pinMode(led, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  int lecture = analogRead(photo);
  Serial.println(lecture);
  if (lecture > 500) {
    digitalWrite(led, LOW);
  }
  if (lecture < 400) {
    digitalWrite(led, HIGH);
  }
}

```

https://www.youtube.com/embed/axW2sedAiUQ

Conclusion:

Cette série d'exercices sur Arduino marque une étape importante dans notre compréhension du processus d'innovation technique. Nous avons progressé d'une exécution binaire simple (allumer/éteindre) vers la conception d'un **système autonome et intelligent** capable de traiter des données extérieures pour agir en conséquence.

La maîtrise de ces outils (capteurs, microcontrôleurs et logique de code) nous offre une base solide pour prototyper des solutions complexes lors de nos futurs projets de Master, en faisant le pont entre le monde numérique et les objets physiques.

Revision #7
Created 16 January 2026 06:57:58 by Rock Anita
Updated 3 May 2026 23:29:26 by Rock Anita