Si vous ne l'avez pas déjà fait lors de l'installation de la Gamebuino META, je vous invite à copier coller le programme dans Arduino et le lancer sur votre console.

```
#include <Gamebuino-Meta.h>

void setup(){
  gb.begin();
}

void loop(){
  while(!gb.update());
  gb.display.clear();

  // C'est ici que le plus gros du programme se déroule
  gb.display.print("hello, world");
}
```

Passons aux choses sérieuses! On va décortiquer ça pour comprendre ce qui se passe.

```
#include <Gamebuino-Meta.h>
```

Vous verrez souvent des **#include** au début des programmes. Ils servent à inclure des programmes externes. Ici, c'est toute la bibliothèque Gamebuino, qui va nous être necéssaire par la suite.

```
void setup(){

}

void loop(){

}
```

Tout programme Gamebuino a deux **fonctions** principales et necéssaires, `setup` et **`loop`**. Toutes les instructions situées entre les **accolades** `{ }` seront exécutées quand la fonction est exécutée.
La **fonction**`setup` est exécutée une fois au démarrage du jeu, puis **`loop`** s'exécute en boucle, indéfiniment... d'où le nom ;)

```
gb.begin();
```

Vous voyez que notre fonction `setup` ne contient qu'une **instruction**, **`gb.begin`**. Elle sert à initialiser la Gamebuino, c'est pour ça qu'il est necéssaire de l'exécuter une fois et une seule, au démarrage. Vous remarquerez que chaque **instruction** du programme est suivi d'un point virgule `;` sans quoi vous aurez une erreur en essayant de lancer le programme.

```
while(!gb.update());
```

Cette instruction est également nécessaire dans tout programme Gamebuino. Elle va s'occuper de tout se qui se passe en arrière plan comme mettre à jour l'écran, regarder si les boutons ont été appuyés ou encore jouer les sons. De plus, elle s'occupe de limiter la vitesse à laquelle **`loop`** va s'exécuter, afin d'avoir une vitesse de

jeu constante. Par défaut, les jeux tournent à 25 images par seconde, ce qui signifie que la fonction **loop** est exécutée toutes les 0,04 secondes. C'est rapide, et c'est un bon compromis pour avoir des animations fluides sans demander trop de puissance de calcul.

```
gb.display.clear();
```

Comme nous exécutons **loop** 25 fois par seconde, il faut effacer le contenu de l'écran avant de commencer de tracer l'image suivante... sinon, ça serai un joli bazar ! C'est justement ce que fait **gb.display.clear**

```
// C'est ici que le plus gros du programme se déroule
```

Les lignes qui commencent par // sont des **commentaires**. Ils sont ignorés par l'ordinateur lors de l'exécution du programme, ils permettent donc d'ajouter des notes personnelles en langage parlé.
Les **commentaires** sont pratiques pour comprendre rapidement comment fonctionne un programme. Pensez à quand vous vous relirez dans un mois ;)

```
gb.display.print("hello, world");
```

Nous avons fait le tour du squelette necéssaire pour faire un programme Gamebuino. On va donc pouvoir passer aux choses sérieuses! gb indique qu'il s'agit d'une fonction qui concerne la Gamebuino, et **gb.display** indique qu'on s'adresse plus particulièrement à l'écran. **gb.display.print** est une fonction qui sert à écrire une **chaîne de caractères** sur l'écran de la gamebuino. Les **chaînes de caractères** sont délimitées par des guillemets " ".
Le language C/C++ que vous êtes en train d'utiliser est **sensible à la casse** (rien à voir avec le fait de faire tomber votre programme par terre). Les majuscules et minuscules ont une importance : print n'est pas la même chose que **Print**. Si vous faites la faute, vous aurez une erreur au moment de compiler le programme!

A VOUS DE JOUER!

Maintenant, vous pouvez écrire ce que vous voulez sur la Gamebuino. Et si vous nous composiez un petit poème ?

Partagez le résultat avec #gamebuino #helloworld pour célébrer votre tout premier programme ! :)

- **Astuce #1** Vous pouvez utiliser \n pour insérer des retour à la ligne dans votre texte.
- **Astuce #2** Vous pouvez changer la couleur du texte avec **gb.display.setColor**. Allez voir Graphics::setColor dans la Référence pour connaître la bonne syntaxe!
- **Astuce #3** Pour changer l'échelle du texte, il y a la fonction Graphics::setFontSize.

EXEMPLE DE SOLUTION

```
#include <Gamebuino-Meta.h>

void setup(){
  gb.begin();
}

void loop(){
  while(!gb.update());
  gb.display.clear();

  //Affichage du message à l'écran
  gb.display.print("'hello, world'\nest une tradition\ninstauree en\n\n");
  //Texte à l'échelle 2
  gb.display.setFontSize(2);
  gb.display.print(1978);
  //Retour à l'échelle 1 et changement de couleur
  gb.display.setFontSize(1);
  gb.display.setColor(BROWN);
  gb.display.print("\n\n\n       - GAMEBUINO");
}
```