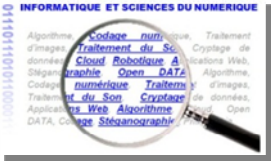


Premiers pas en Réalité augmentée

Thème d'étude : Conception d'une vidéo en réalité augmentée
Nature de la production finale attendue : Conception d'une vidéo en réalité augmentée générée par le logiciel Processing. Cette vidéo doit présenter une visite partielle d'une salle de cours avec incrustation à partir de marqueurs d'images générées par du code.
Cahier des charges : Cette vidéo comporte 5 marqueurs.

On donne :

- *Le programme partiel avec détection à partir de 3 marqueurs.*
- *Les 3 marqueurs déjà paramétrés*

On demande :

- *Adapter le programme existant pour ajouter la gestion des marqueurs 4 et 5 pour faire apparaitre un cube, puis une sphère.*

On exige :

- *Un programme fonctionnel en 2 étapes*
  - *Marqueur 4 tout d'abord avec validation par l'enseignant*
  - *Marqueur 4 et 5 avec validation par l'enseignant*

Conditions	Travail dirigé Durée : 3 h	**Moyens**	• Poste informatique sous Windows • Logiciels PROCESSING
Prérequis	Représentation de l'information	**Niveau**	Classe de Terminale S spécialité ISN
Compétences	• C1.1 Justifier dans une situation donnée, un codage numérique ou l'usage d'un format approprié, qu'un programme réalise l'action attendue... • C2.3 Développer une interface logicielle ou une interface homme-machine, un algorithme, un programme, un document ou fichier numérique...		

Eléments du programme	• Représentation de l'information
	  o Structuration et organisation de l'information
	• Algorithmes
	  o Algorithmes simples		

Table des matières :

1. Les fonctions principales

Le programme utilise un certain nombre de fonctions dont certaines reviennent assez souvent :

- import : permet l'importation de librairie
- void : indique que la fonction ne renvoie aucune valeur
- println : affiche dans la zone de texte l'évolution d'une, ou plusieurs variables
- text : fonction d'affichage de texte
- loadImage : fonction de chargement d'une image (appelle une image)
- fill (R,G,B) : permet le changement de couleur d'une fonction telle que « text »

2. Le programme commenté

Importation des librairies (*java, opengl, nyartoolkit, gsvideo*), chaque librairie permet d'ajouter des fonctionnalités, le programme peut utiliser plus ou moins de fonction selon les librairies importées.

```
import java.io.*;
import processing.opengl.*;
import jp.nyatla.nyar4psg.*;
import codeanticode.gsvideo.*;
```

Déclaration des variables, il y a autant de variable que de pattern à appeler.

```
//initialisation des variables
GSCapture cam;     //variable pour le flux vidéo
MultiMarker nya_1; //variable 1
MultiMarker nya_2; //variable 2
MultiMarker nya_3; //variable 3
MultiMarker nya_4; //variable 4
MultiMarker nya_5; //variable 5
int fps = 30;     // défini le nombre de fps (frame per second)
GSMovieMaker  mm; //variable pour l'enregistrement du flux vidéo
```

Void devant la fonction définit le fait qu'elle ne renvoie pas de valeur. La fonction setup( ) sert à indiquer les options par défaut tel que la définition du cadre vidéo (sa taille), le lancement de la vidéo, de l'enregistrement vidéo et la détection des marqueurs.

```
void setup() {
 // enregistrement
 mm = new GSMovieMaker (this, width, height, "drawingl.ogg", GSMovieMaker.THEORA, GSMovieMaker.MEDIUM, fps);
 mm.start();   //lance l'enregistrement
 size(640,480,P3D); //définition de la taille de la fenêtre d'affichage
 colorMode(RGB, 100); //définit le mode de couleur interprêté par processing, en RGB ici
 println(MultiMarker.VERSION);  //affiche dans la zone de texte l'évolution d'un type de variable
 cam=new GSCapture(this,640,480);  //définit la taille de capture du flux
 cam.start();  //lance l'aquisition du flux vidéo
```

Déclaration des marqueurs, chaque variable est associée à un fichier du pattern, ainsi la variable pourra reconnaître le marker grâce au fichier *.patt* appelé. Par exemple pour la variable *nya_1* on l'associe au *marker1.patt*, ainsi il peut lire et reconnaître le marker 1.

Note : Il faut placer ce fichier marker1.patt dans le répertoire « data »

```
//déclaration des markers, on associe un marker à un pattern
 //marker1
 nya_1=new MultiMarker(this,width,height,"camera_para.dat",new NyAR4PsgConfig(NyAR4PsgConfig.CS_LEFT_HAND,NyAR4PsgConfig.TM_NYARTK));
 nya_1.addARMarker("marker1.patt",80);

 //marker2
 nya_2=new MultiMarker(this,width,height,"camera_para.dat",new NyAR4PsgConfig(NyAR4PsgConfig.CS_RIGHT_HAND,NyAR4PsgConfig.TM_NYARTK));
 nya_2.addARMarker("marker2.patt",80);

 //marker3
 nya_3=new MultiMarker(this,width,height,"camera_para.dat",new NyAR4PsgConfig(NyAR4PsgConfig.CS_LEFT_HAND,NyAR4PsgConfig.TM_NYARTK));
 nya_3.addARMarker("marker3.patt",80);
```

La fonction drawgrid( ) permet d'établir un quadrillage avec des informations liées au dessin. On peut ainsi définir les couleurs et la taille des lignes et des bordures ainsi que sauvegarder et réinitialiser les changements de coordonnées.

```
void drawgrid() //implemente un quadrillage
{
 pushMatrix(); //sauvegarde les changements de coordonnées
 stroke(0); //définit la couleur utilisée pour dessiner des lignes et des bordures autour des formes (en RGB de 0 à 255)
 strokeWeight(2); //définit la taille des lignes et des bordures autour des formes
 popMatrix(); //réinitialise les changements de coordonées
}
```

La fonction background permet de changer la couleur du fond de la fenêtre Processing.

```
  //initialisation des markers à détecter
  cam.read();
  image(cam, 0, 0, 640, 480);
  nya_1.detect(cam);  //marker1
  nya_2.detect(cam);  //marker2
  nya_3.detect(cam);  //marker3
```

On peut alléger, simplifier le programme en définissant une variable contenant les informations pour les zones de texte (PFont) de la manière suivante.
Ainsi on obtient une variable « w » qui contient ici les informations suivantes :

-nom de police : Comic sans MS
-taille de police : 35

On lui associe ici ces valeurs grâce à la fonction createFont( ).
Il n'y a plus qu'à utiliser la variable comme valeur de la fonction « textFont » pour appeler ces paramètres ci.

```
//initialisation d'une police de texte

PFont w;
w = createFont("Comic sans MS",35,true);
```

Une fonction if( ) contenant les informations à lire par le programme est utilisée pour chaque marker, chacun des markers étant appelés par leur variable (nya_1 en l'occurrence pour le marqueur 1). Ici si le marqueur est détecté, le texte « Bienvenue » s'affiche grâce à la fonction text( ) contenant aussi les coordonnées de l'affichage

```
//marker 1
 if((nya_1.isExistMarker(0))){ //fonction "si", si le marqueur 1 est présent executé la suite.
  nya_1.beginTransform(0); //début des modifications virtuelles
   fill(0,0,255); //permet le changement de couleur de fonction (r,g,b)
   drawgrid();
   rotate(0); //permet la rotation avec une valeur angulaire
   textFont(w);  //utilisation de la police de texte défini plus haut
   text("Bienvenue", -50, 0);   //afficher le texte "Bienvenue", de type text(".....",x,y)
   nya_1.endTransform(); //fin des modifications virtuelles
}
```

On retrouve donc la fonction if à chaque marker, ici on définit une nouvelle variable de type image grâce à la fonction PImage suivi du nom de la variable (sti2d dans ce cas) on appelle la variable en l'associant à une image, la fonction loadImage(«...») permet de lier la variable à l'image spécifiée (elle doit se trouver dans le même dossier que le fichier, dans le cas contraire il faut spécifier son chemin depuis le fichier processing).
On affiche l'image grâce à la fonction image( ) en lui spécifiant le nom de la variable à utiliser ainsi que sa taille et ses coordonnées.

```
//marker 3
    if((nya_3.isExistMarker(0))){
    nya_3.beginTransform(0);
    PImage sti2d; //déclaration d'une image
    sti2d = loadImage("sti2d.jpg");  //charger l'image "sti2d.jpg" et l'associe à la variable "sti2d"
    image(sti2d,-50,-25,100,50); //appel l'image (image, coord-X, ccord-Y, hauteur, largeur)
    fill(0,0,255);
    drawgrid();
    rotate(0);
    nya_3.endTransform();

 }
```

La partie suivante permet d'enregistrer le flux vidéo, le fichier vidéo étant nommé et définit plus haut dans le programme (lors des déclarations de variables). L'enregistrement se fait au format .ogg. La fonction keyPressed( ) permet d'utiliser une touche pour mettre fin a l'enregistrement et au programme.

```
loadPixels(); // charge les pixels pour enregistrer le flux vidéo
 mm.addFrame(pixels); // ajoute les images au fichier vidéo depuis les pixels chargés

void keyPressed() // fonction qui permet d'utiliser le clavier
{
  if (key == ' ') // associe la fonction suivante à la barre d'espace
  {
    mm.finish(); //stoppe l'enregistrement si la barre d'espace est appuyée
   exit(); // quitte le programme une fois le fichier écrit
  }
} //fin du programme
```

Ainsi si on lit le code on remarque la structure suivante (en la simplifiant) :

Si (la touche espace est appuyé)
mettre fin à l'enregistrement
fermer le programme

3. Modification du programme – travail demandé

3.1 Charger sur Processing le fichier partiel isn_élève fourni, analyser la constitution du code utilisé pour l'affichage d'animation à partir des marqueurs 1 à 3.

3.2 Compléter ce programme (voir ci-dessous) en écrivant les lignes de code permettant à deux autres marqueurs, que l'on nommera marqueurs 4 et 5, d'afficher un cube de couleur verte non-opaque de côté et une sphère.

```
//marker 3
    if((nya_3.isExistMarker(0))){
    nya_3.beginTransform(0);
    PImage sti2d; //déclaration d'une image
    sti2d = loadImage("sti2d.jpg");  //charger l'image "sti2d.jpg" et l'associe à la
    image(sti2d,-50,-25,100,50); //appel l'image (image, coord-X, ccord-Y, hauteur,
    fill(0,0,255);
    drawgrid();
    rotate(0);
    nya_3.endTransform();

 }

//marker 4

//à compléter...

//marker5

//à compléter...
```

3.3 Enregistrer une vidéo et vérifier le bon fonctionnement de votre programme.

Correction :

```
//marker4

 nya_4=new MultiMarker(this,width,height,"camera_para.dat",new NyAR4PsgConfig(NyAR4PsgConfig.CS_RIGHT_HAND,NyAR4PsgConfig.TM_NYARTK));
 nya_4.addARMarker("marker4.patt",80);

//marker5
 nya_5=new MultiMarker(this,width,height,"camera_para.dat",new NyAR4PsgConfig(NyAR4PsgConfig.CS_RIGHT_HAND,NyAR4PsgConfig.TM_NYARTK));
 nya_5.addARMarker("marker5.patt",80);
}
```

```
//initialisation des markers à détecter
cam.read(); // lance la lecture du flux
image(cam, 0, 0, 640, 480);
nya_1.detect(cam);  // détection du marker1
nya_2.detect(cam);  // détection du marker2
nya_3.detect(cam);  // détection du marker3
nya_4.detect(cam);  // détection du marker4
nya_5.detect(cam);  // détection du marker5
```

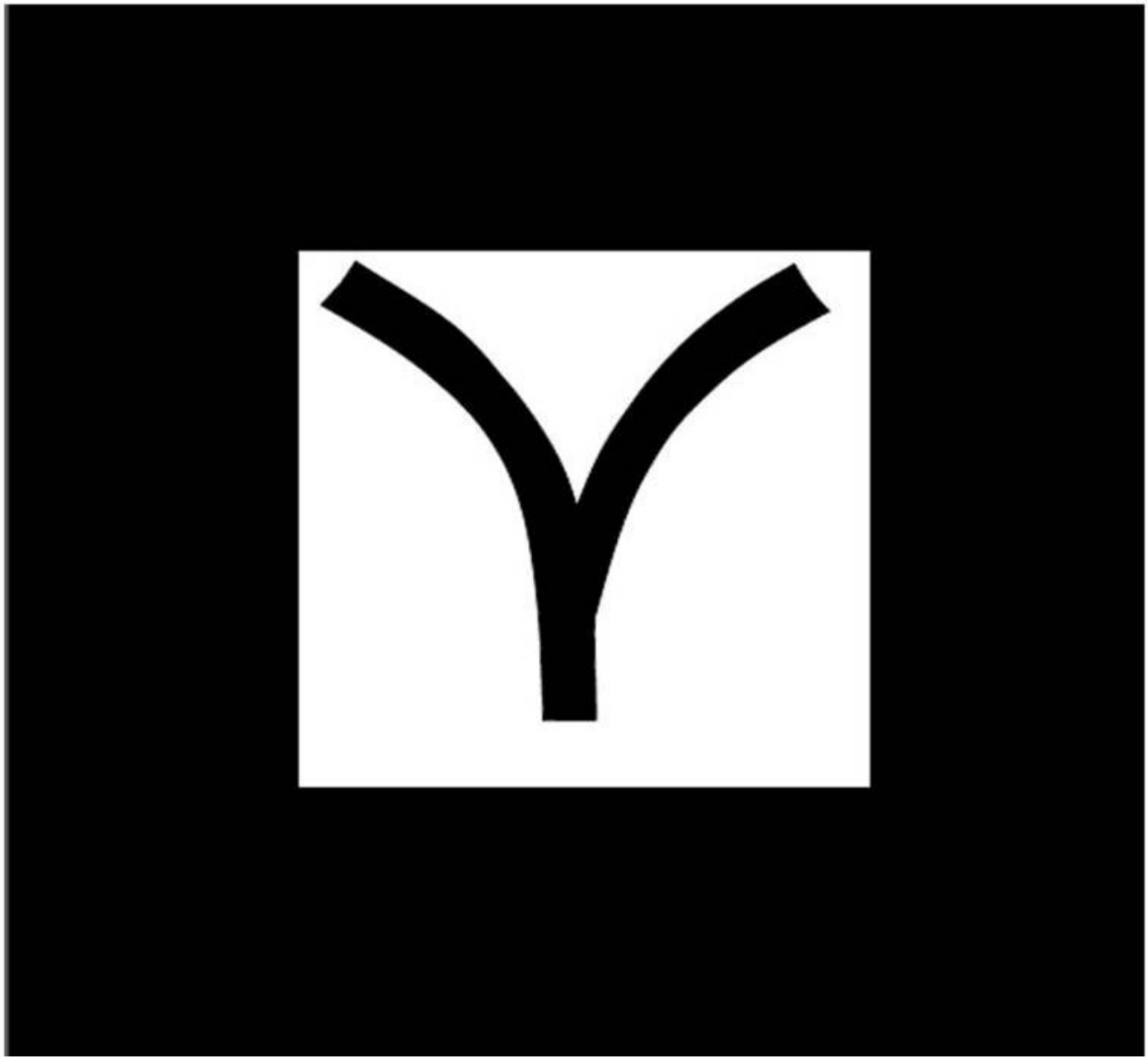
```
//marker 4
if((nya_4.isExistMarker(0))){
    nya_4.beginTransform(0);
    fill(0,255,0,100);
    drawgrid();
    translate(0,0,20); //définit la place de l'objet dans le repère (x,y,z) en fonction de la fenêtre
    rotate((float)c/100); // angle défini en radian par le biais de la fonction de float
    box(40); //affichage d'un cube de taille (x)
    nya_4.endTransform();
 }

//marker5
 if((nya_5.isExistMarker(0))){
    nya_5.beginTransform(0);
    fill(0,255,255);
    drawgrid();
    translate(0,0,20);
    rotate((float)c/100);
    sphere(40); //affichage d'une sphère de taille (x)
    nya_5.endTransform();
 }
```

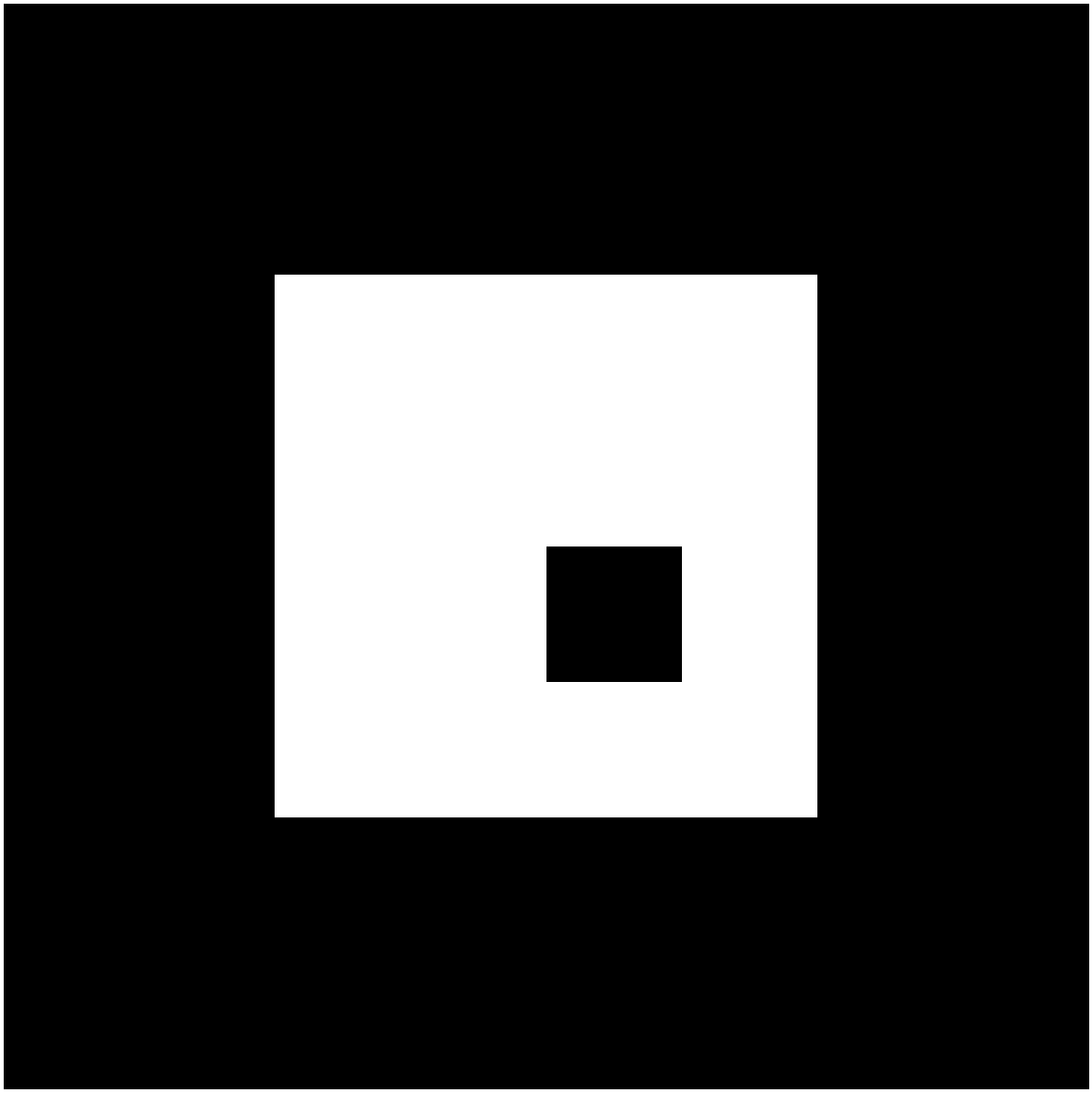
4. Les marqueurs utilisés

marker 1

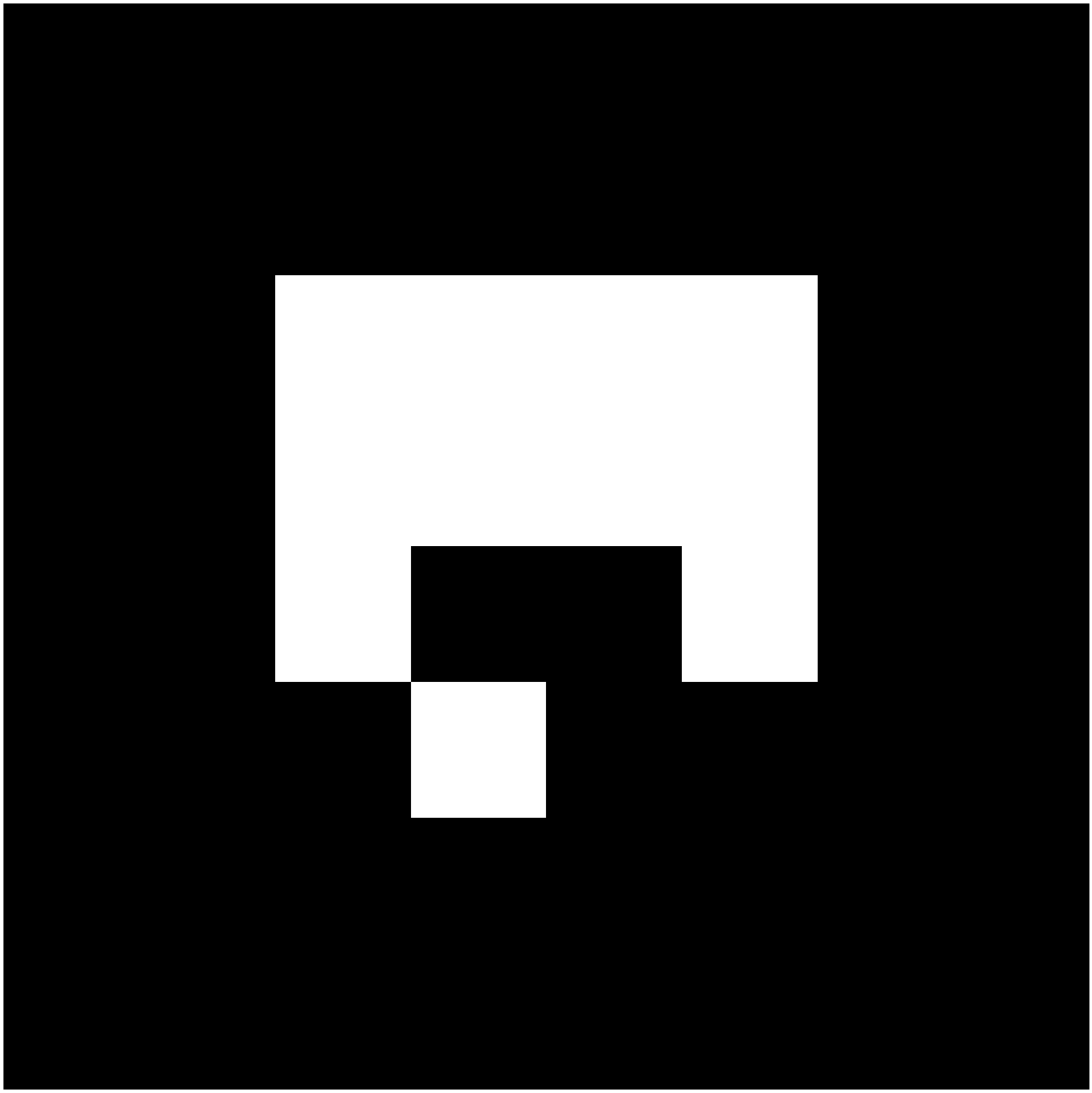
marker 2



marker 3



marker 4



marker 5