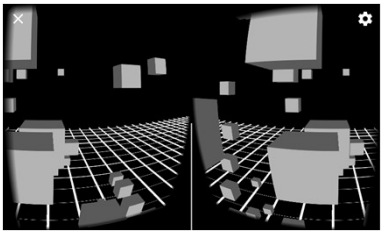
IX – Réalité Immersive avec Processing et Android

141

IX.1 – Réalité Virtuelle immersive

- Objectif :
  - Rendre les scènes visibles en stéréo 3D
  - et prendre en compte des mouvements de l'utilisateur
    - pas aussi simple qu'un film stéréo
- Principe de l'immersion :
  - Utilisation d'un casque (avec lentilles)



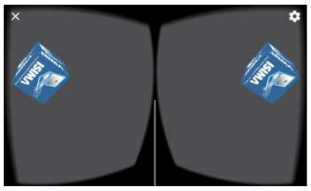
  - Exploitation d'un smartphone :
    - processeur (traitements informatiques nécessaires) exécutant du Java / Android
    - avec écran, partagé pour avoir des images différentes sur chaque œil
    - avec capteurs de positionnement (accéléromètre et gyroscope)
    - et éventuellement, en utilisant une caméra (RA ou RE)

142

IX.1 – Ecriture d'applications *VR* pour smartphone Android

- Modification de l'application pour créer une image adaptée
  - Processing en « Mode Android », et application de type « VR »
  - Ajout de la librairie VR par import dans le code Processing
    - `import processing.vr.*; // Gestion du mode VR...`
  - Remplacement de `size()` par `fullScreen()` en MONO ou STEREO (VR)
    - `fullScreen(MONO)` `// Ouverture en mode 3D, avec un rendu 'presque' comme sur une App normale... car le cube est localise dans l'espace, donc, il faut orienter correctement le smartphone pour le voir !`

    - `fullScreen(STEREO)` `// ou fullScreen(VR)` `Ouverture en mode 3D, avec deformation lenticulaire`

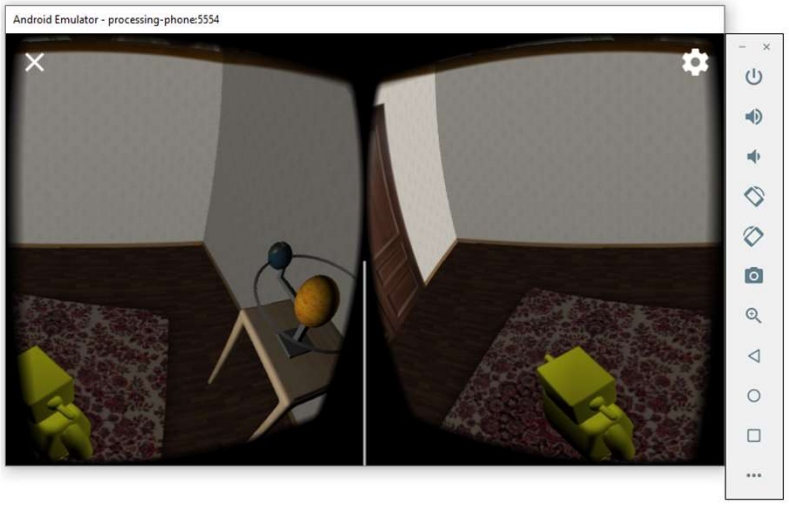


143

IX.2 – Exemple 14 : applications *VR* immersive

- Exemple « d'instructables » :

https://www.instructables.com/id/Mobile-Virtual-Reality-Using-Processing-for-Androi/



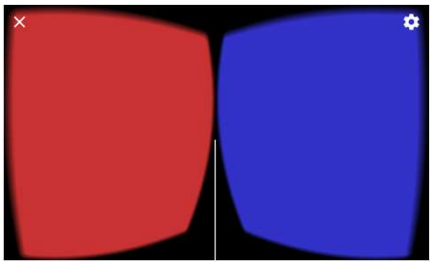
144

IX.3 – Programmation VR avancée : écrans séparés

- Possibilité de gérer chaque image séparément !
  - `draw` est appelée 2 fois par frame (!), car une fois pour chaque œil...
    - présence d'une nouvelle fonction « `calculate` » pour les traitements à ne faire qu'une fois
  - récupération du graphique `PGraphicsVR` et lecture de la variable `eyeType`
    - PVR.LEFT et PVR.RIGHT, ou PVR.MONOCULAR

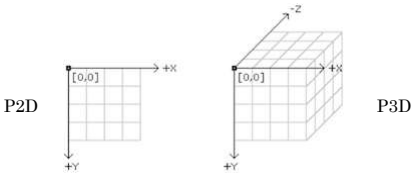
```
void setup() { // Details ici :  https://android.processing.org/tutorials/vr_intro/index.html
  fullScreen(STEREO); // En mode stereo
}
void draw() {  // En boucle, 2 fois par frame
  PGraphicsVR pvr = (PGraphicsVR)g;

  if (pvr.eyeType == PVR.LEFT) {
    background(200, 50, 50); // En rouge
  } else if (pvr.eyeType == PVR.RIGHT) {
    background(50, 50, 200); // En bleu
  } else if (pvr.eyeType == PVR.MONOCULAR) {
    background(50, 200, 50); // En vert
  }
}
void calculate() {  // En boucle, 1 fois par frame : Nouvelle fonction specifique mode VR...
}
```

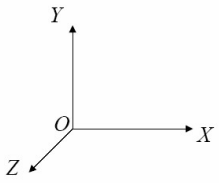


IX.4 – Programmation VR avancée : Référentiel

- Changement du référentiel Processing 3D en VR
  - Rappel sur le système de coordonnées 3D « standard »
    - par défaut : origine (0,0,0) en haut et à gauche, Y pointant en bas
    - beaucoup de `translate(width/2, height/2, 0);  // Positionne au centre`

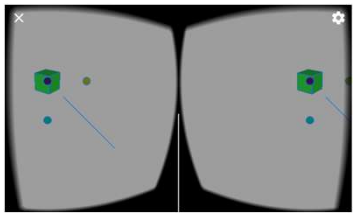


  - Changement du référentiel par appel à `CameraUp()` dans le `setup` :
    - origine (0,0,0) au **centre** de la scène,
    - Y **pointant vers le haut**
    - plus naturel pour une scène 3D, et évite donc beaucoup de translate...

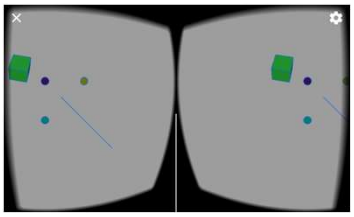


IX.5 – Programmation VR avancée : GUI

- Création d'un GUI : « Graphical User Interface »
  - donc, information placée au premier plan sur l'écran,
  - et non interactif (pas de clic possible)
- Changement du référentiel pour GUI !
  - Passage en écriture sur « écran » en changeant le point de vue utilisateur
  - Une nouvelle fois un changement de repère
    - Utilisation du « Point de vue de l'utilisateur » : fonction `eye()`
    - Appel à la fonction, puis écriture des commandes classiques de tracé
  - Attention : retour au référentiel initial
    - utilisation de `pushMatrix()` et `popMatrix()` qui encadrent les instructions de tracé



3 cercles et une ligne en GUI



même scène après rotation x et y
GUI inchangée

147

IX.5 – Programmation VR avancée : Capteurs par API Sensor

- Utilisation de l'API Sensor de l'Android SDK
  - Ouverture d'un gestionnaire, sur le `context` de l'`activity` :

```
import android.content.Context; // Import du context
import android.hardware.Sensor; // Import des capteurs
import android.hardware.SensorManager; // Import du gestionnaire
```

  - Mise en marche du gestionnaire sur les capteurs, dans le `setup` :

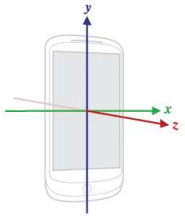
```
Context context;     SensorManager manager;     Sensor sensor;
void setup() {
  context = getActivity();
  manager = (SensorManager)context.getSystemService(Context.SENSOR_SERVICE);
  sensor = manager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);}
```

  - Ajout d'un listener :

```
import android.hardware.SensorEvent; // Import des evenements capteurs
import android.hardware.SensorListener; // Import du listener
AccelerometerListener listener;
listener = new AccelerometerListener();
manager.registerListener(listener, sensor, SensorManager.SENSOR_DELAY_GAME);
```

  - Puis lecture :

```
class AccelerometerListener implements SensorEventListener {
  public void onSensorChanged(SensorEvent event) {
    ax = event.values[0];    ay = event.values[1];    az = event.values[2];
  }
  public void onAccuracyChanged(Sensor sensor, int accuracy) {
  }
}
```



https://android.processing.org/tutorials/sensors/index.html

148

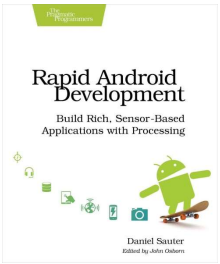
IX.5 – Programmation VR avancée : Bluetooth de Ketai

- Utilisation pour les « manettes » et « contrôleurs » :

ACGAM R1

MOCUTE

- Bluetooth par Ketai :
  - Attention : utilisation de « Open Sound Control » pour l'envoi de message sur le réseau (http://www.sojamo.de/libraries/oscP5/)
  - Découverte, connexion et appariement
  - Lecture des données
- Livre gratuit :
  - Daniel Sauter, « Rapid Android Development »
  - en ligne : https://www.mobileprocessing.org/p2p.html



149

TD 11 Application Android VR avec GUI

- Travail demandé :
  - Faire une application immersive (au format « VR ») qui intègre une GUI
    - Mettre Nom et prénom
    - et un petit logo ISIMA
  - Intégrer ensuite cette GUI sur un petit exemple :
    https://www.instructables.com/id/Mobile-Virtual-Reality-Using-Processing-for-Androi/

150