

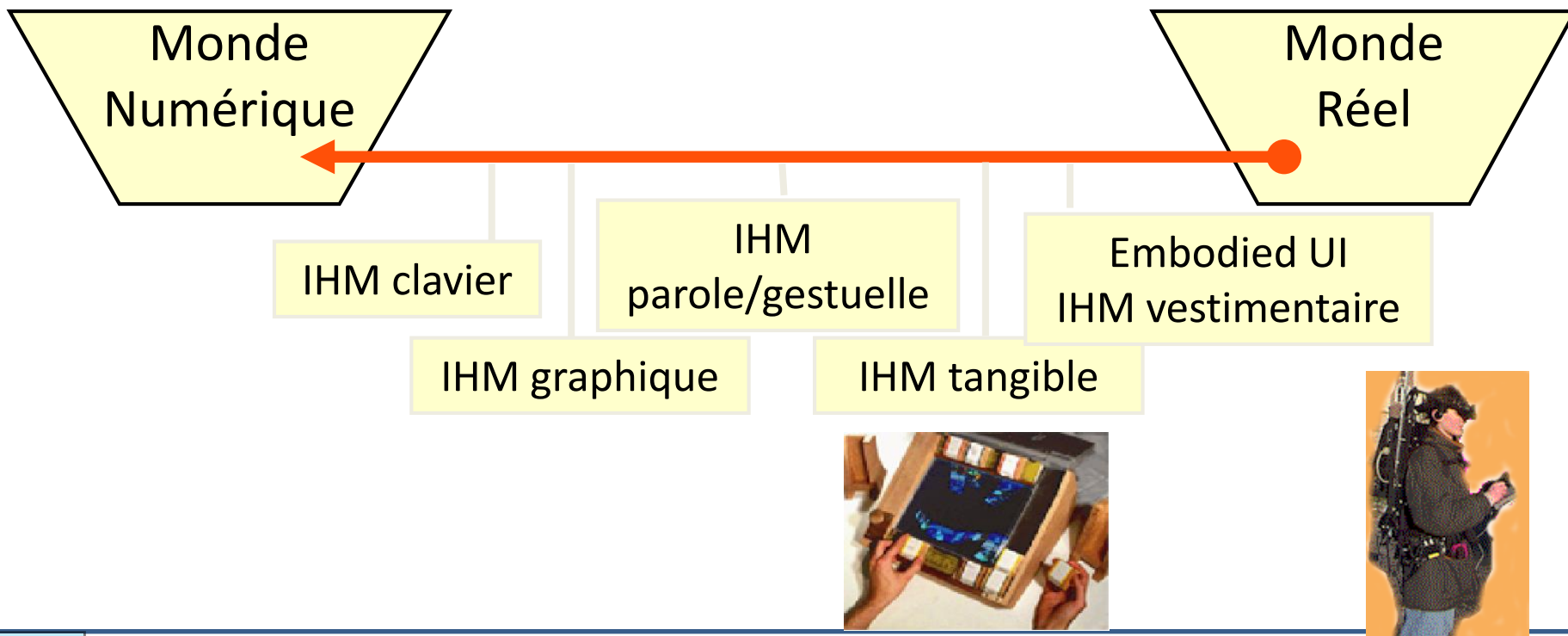
Systemes Mixtes : Réalité Augmentée et Virtualité Augmentée

Philippe.Renevier@unice.fr

<http://atelierihm.unice.fr/enseignements/techniques-interaction>

Virtualité Augmentée

- Limitation de la souris
- Interagir plus naturellement
- Continuum de Milgram



Systemes Mixtes

- Profusion de termes

- Bit / Atome

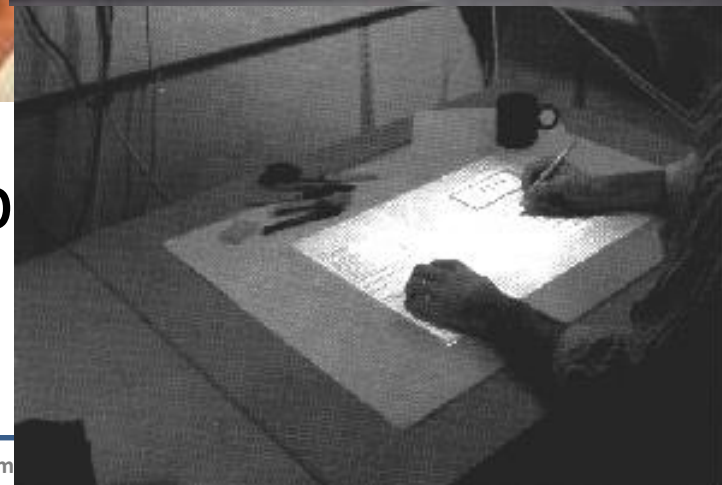
- Réalité augmentée

- Réalité augmentée

- (**mobile**)

- ...

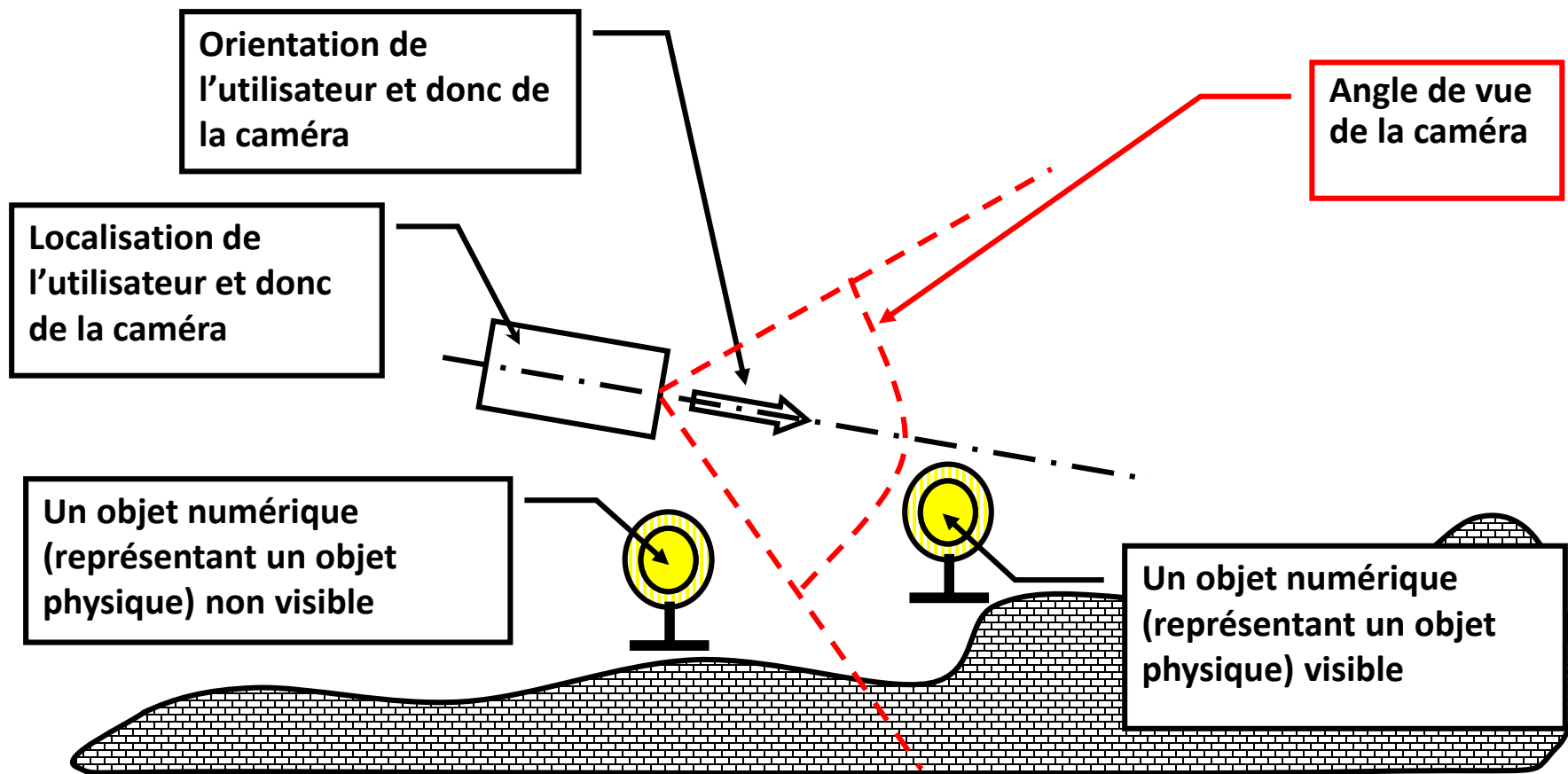
- Un but : combiner les entités p
numériques



Systemes Mixtes

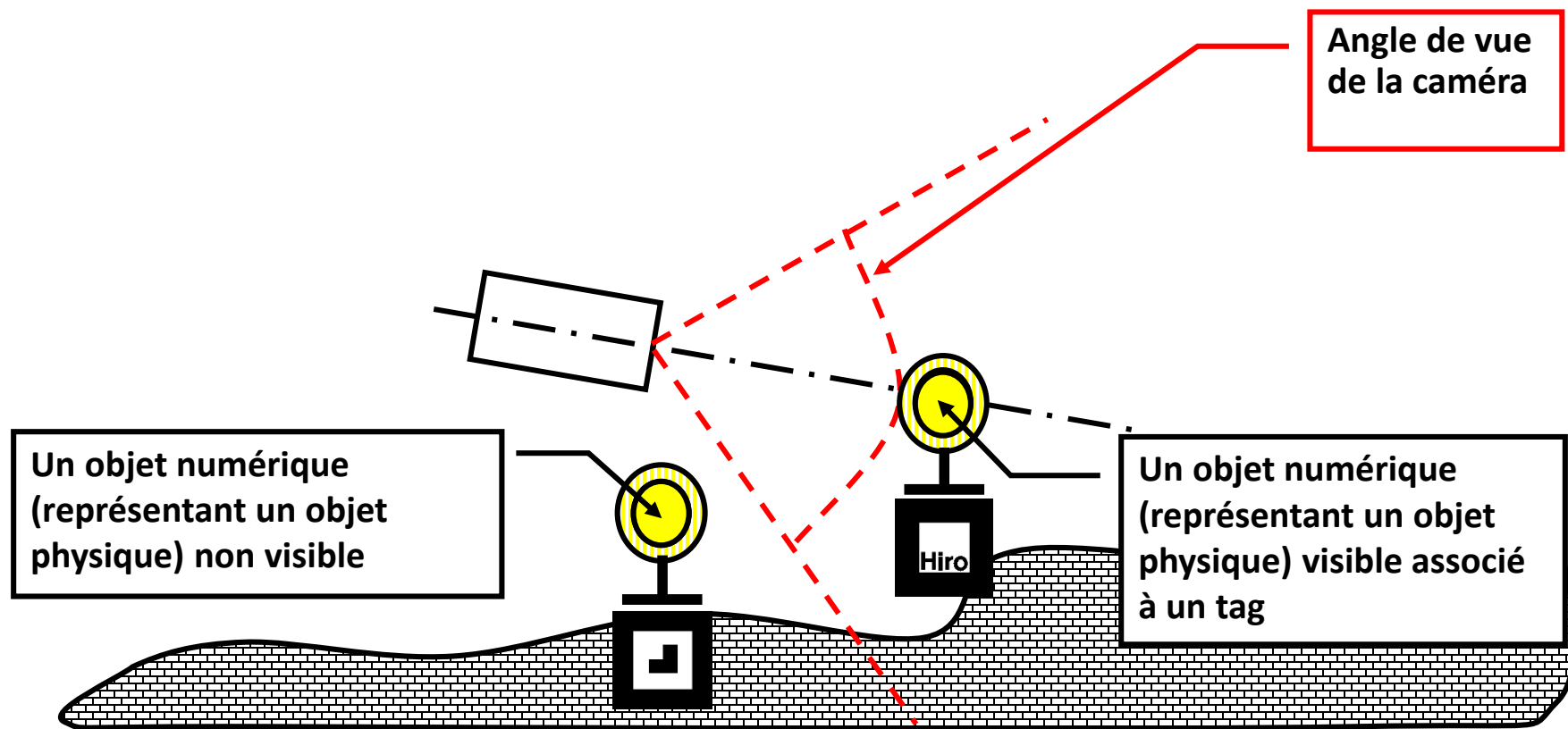
- Mélange d'activité dans le monde physique et dans le monde numérique
- S'affranchir de contraintes historiques d'interaction
- Réalité Augmentée = aller sur le terrain

Géolocalisation



Steven Feiner et la Touring Machine

Reconnaissance de Tag/Image



Jun Rekimoto et la NaviCam , puis l'ARToolkit

Affichage

- Affichage via un dispositif semi-transparent
- Affichage par projection
 - c.f. Parnav Mistry et son projet 6thsense
- Affichage sur le flux vidéo
 - ARToolkit

Interactions

- Dispositifs « portables »
- Gestes
 - Mains « nues »
 - Mains « équipées »
 - Déplacements d'objets reconnus
- Objets communicants

Caractéristiques des augmentations

| | | | |
|---|--------------------|---------------|----------------------|
| Cible de l'augmentation | Utilisateur | Objets | Environnement |
| | | | |
| Type de l'augmentation | Evaluation | | Exécution |
| | | | |
| Temporalité de l'augmentation | Persistent | | Ephémère |
| | | | |
| Mode d'interaction lors de la création de l'augmentation | Actif | | Passif |
| | | | |
| Mode d'interaction lors de la modification de l'augmentation | Actif | | Passif |
| | | | |

Références des exemples

- *NaviCam* : Rekimoto, Nagao. The World through the Computer : Computer Augmented Interaction with Real World Environments. In Proceedings of UIST'95, Pittsburgh, 1995, pp 29-36.
- *Mediablocks* : Ishii, Ullmer, "Tangible Bits : Towards Seamless Interfaces between People, Bits and Atoms", In Proceedings of the ACM conference CHI'97, Atlanta, March 1997. pp 234-241.
- *Phicons* : Want, Fishkin, Gujar, Harrison. Bridging Physical and Virtual Worlds with Electronic Tags. In Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 99), Pittsburgh, May 1999, pp 370-377.
- *Digital Desk* : Wellner. The Digital Desk calculator : tangible manipulation on a desk top display. In Proceedings of the fourth annual ACM symposium on User interface software and technology, November 1991, pp 27-33.

Exemple de caractéristiques

| | | | |
|---|---|----------------|------------------------------|
| Cible de l'augmentation | Utilisateur | Objets | Environnement |
| | <i>Navicam</i> | <i>Phicons</i> | <i>Digital Desk</i> |
| Type de l'augmentation | Evaluation | | Exécution |
| | <i>Navicam</i> | | <i>Phicons, Digital Desk</i> |
| Temporalité de l'augmentation | Persistent | | Ephémère |
| | <i>Navicam, Phicons</i> | | <i>Phicons, Digital Desk</i> |
| Mode d'interaction lors de la création de l'augmentation | Actif | | Passif |
| | <i>Phicons, Digital Desk</i> | | <i>Navicam</i> |
| Mode d'interaction lors de la modification de l'augmentation | Actif | | Passif |
| | <i>NaviCam (configuration par l'utilisateur), Phicons, Digital Desk</i> | | |

Développements

- Nature mobile des activités



4G
Ready



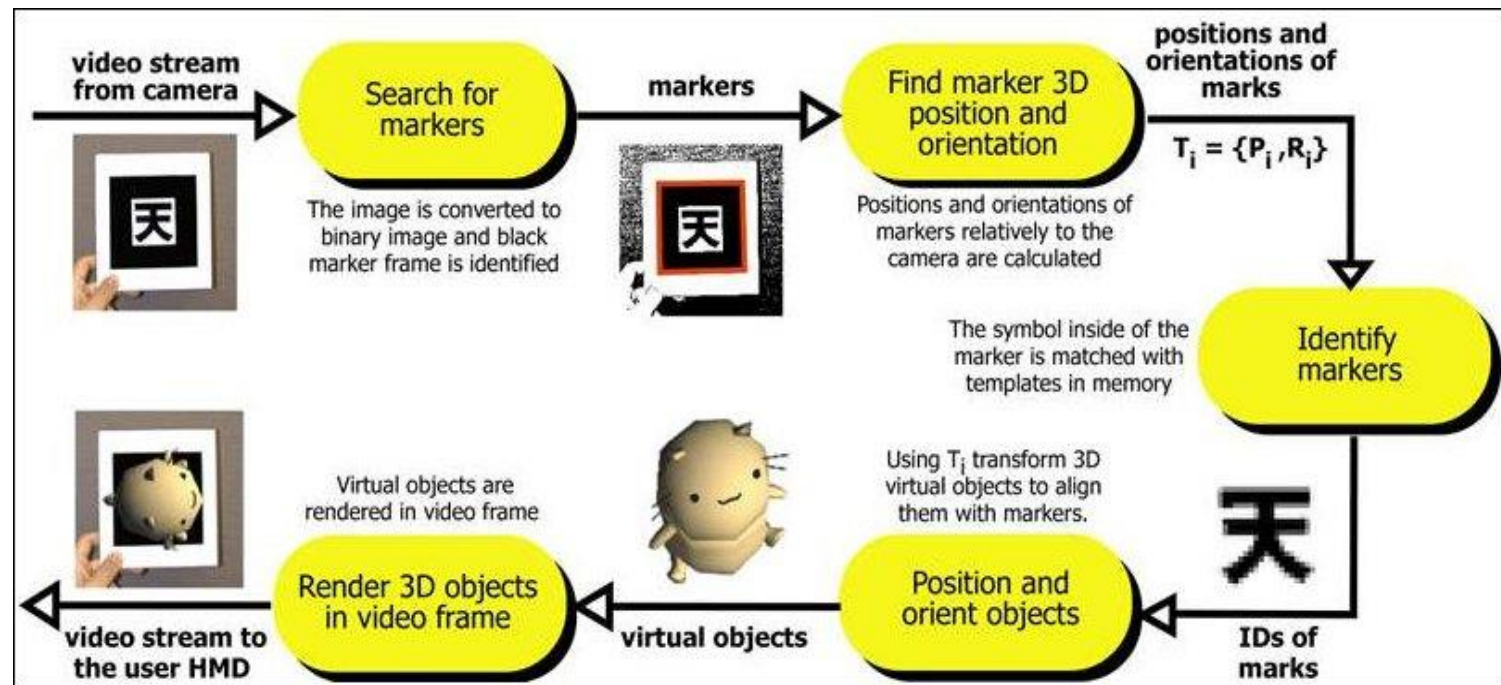
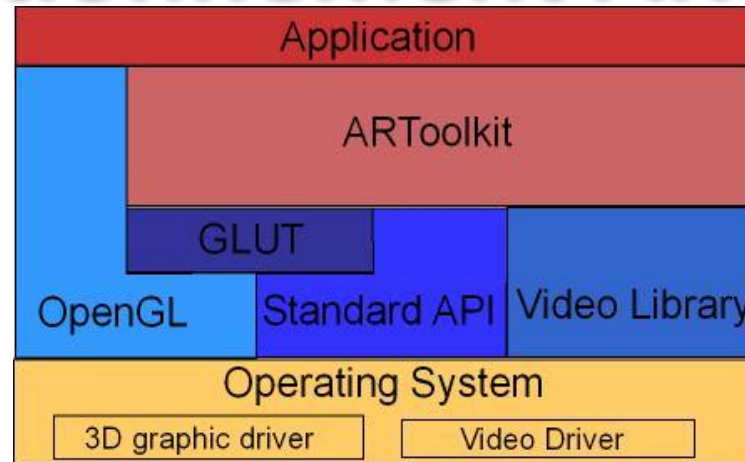
Virtual showcase...

- Nature collaboratif
- Progrès technologiques
- Manque de stabilité...
mais de plus en plus de librairies...

ARToolkit

- **ARToolkit** : années 1999 - 2003, H. Kato, M. Billinghurst, en C++
<http://www.hitl.washington.edu/artoolkit/>
- **OpenGL** : <http://www.opengl.org/sdk/docs/man/>
Et <http://www.opengl.org/resources/faq/technical/>
- **GLUT** :
<http://www.opengl.org/resources/libraries/glut/spec3/spec3.html>
- **En java**
 - Une interface pour OpenGL, par exemple jogl :
<http://download.java.net/media/jogl/jogl-2.x-docs/index.html>
 - Une implémentation : **NyARToolkit** (en japonais ☹) :
<http://nyatla.jp/nyartoolkit/wiki/index.php?FrontPage.en>
et <http://sixwish.jp/Nyartoolkit/Java/section01.en/>

Fonctionnement ARToolkit



RA / VA Sur Android / Mobile

- Magic Lens
 - Bier, Stone, Pier, Buxton, DeRose. Toolglass and Magic Lenses: The See-Through Interface. In Conference Proceedings of Siggraph'93, Anaheim, Computer Graphics Annual Conference Series, ACM (1993) 73-80
- Géolocalisation
 - Limitée par la précision du GPS
- Tag
 - Limitée par la robustesse de reconnaissance
 - De plus en plus performant (c.f. Vuforia)

Reconnaissance de Tag

- Différentes (anciennes) implémentations de l'ARToolkit
 - NyaARToolkit
<http://nyatla.jp/nyartoolkit/wiki/index.php?FrontPage.en>
 - AndAR : <http://code.google.com/p/andar/>
- Difficulté de l'OpenGL sous android
 - Possible de retrouver des implémentations de GLUT...
<http://code.google.com/p/andar/source/browse/trunk/AndARpong/src/edu/dhbw/andar/pingpong/GLUT.java>
 - Il y a des portages : <http://jogamp.org/> (ex-jogl)
- Aspect géométrique
 - matrice de transformation...

Géolocalisation via framework sur Android : Layar

- Développement de « couche »
- Ex: Layar : <http://www.layar.com/>
 - Application dédiée et unique
 - compte développeur
 - nouveau calque sur le site Layar
 - Web Service (et BD) pour les POIs (techno JSON)
 - Exemples de point :
 - Bâtiment Templiers: (43.615444, 7.071869)
 - Restaurant Hélios : (43.616586,7.070255)
 - Publication du calque

Géolocalisation via framework sur Android : Wikitude

- <http://www.wikitude.org/developers>
 - <http://www.wikitude.com/app/>
- Évolution au cours du temps :
 - Avant version java (avec création de POI en java)
 - Maintenant avec html / javascript (JSON)
 - + reconnaissance image
- Inscription sur <http://www.wikitude.com>
 - On peut obtenir le sdk
 - <http://www.wikitude.com/developer/documentation/android>

Reconnaissance par Caméra : Vuforia

- *c.f. d'autres : Metaio ; Total Immersion D'Fusion ; etc.*
- Disponibles sur plusieurs plateformes
- Exemples fournis par Vuforia
 - Reconnaissance d'images
 - De boites
 - De tags
 - De texte
 - Des boutons virtuels
 - Adaptation aux HMD

Intégrer Vuforia dans Android Studio

- S'enregistrer sur <https://developer.vuforia.com/>
 - Création d'images reconnaissables en ligne (.dat et xml)
- Télécharger le sdk, les exemples
- Faire un jar nommé armeabi.jar (en ligne de commande ou zippé puis renommé en .jar) contenant un fichier lib/armeabi/libVuforia.so
- Placer armeabi.jar et Vuforia.jar dans le dossiers libs sous app.
 - En principes, dans
File > Project Structure > Dependencies
il y a la règle {dir=libs,include=[*.jar]}

Utiliser Vuforia : les exemples

- Dans le dossier sample, il y a le dossier media avec les ressources (images, pdf)
 - Pour l'exemple : une application qui en lance plusieurs.
 - Mais pas sous la forme d'un projet android
- Créez un projet android vide (dans android studio)
- Recopiez les dossiers assets, media, res dans app/src/main
- Dans le dossier app/src/main/java de votre projet recopiez le contenu de src de l'exemple
- Recopiez aussi le manifest.
- N'oubliez les libs (c.f. transparent précédent)
- Il faut une clef (à obtenir sur le site) à placer dans SampleApplicationSession.java
 - <https://developer.vuforia.com/targetmanager/licenseManager/licenseListing>
 - `Vuforia.setInitParameters(mActivity, mVuforiaFlags, "la clef");`

Exemple : ImageTargets

- Utilisation de classes communes à tous les exemples
 - package `com.qualcomm.vuforia.samples.SampleApplication`
 - package `com.qualcomm.vuforia.samples.SampleApplication.utils`
- Deux classes du package `com.qualcomm.vuforia.samples.VuforiaSamples.app.ImageTargets`
 - `ImageTargets` : l'activité
 - Gère le menu (et les gestures)
 - Lance la réalité augmentée : charger les images reconnus (.xml et .dat qui sont dans `app/assets`) ; configurer la caméra ; lancer la scène openGL et construire l'interface (définie à la main avec `addview`)
 - Gère les pauses / arrêts / retours sur l'application (ne pas monopoliser la caméra... et le traitement de fond)
 - `ImageTargetRender`
 - Implémente la 3D / GL (open Graphics Library)
 - Traite la reconnaissance au moment de se dessiner (`onDrawFrame` -> `renderFrame`)
 - `State state = mRenderer.begin();`
permet de savoir ce qui reconnu (l'objet State encapsule)
 - `mRenderer.drawVideoBackground();`
permet de dessiner l'image de la caméra
 - Tourne sur le thread GL

Traitement d'un résultat (renderFrame)

```
GLS20.glClear(GLS20.GL_COLOR_BUFFER_BIT | GLS20.GL_DEPTH_BUFFER_BIT);
```

```
State state = mRenderer.begin();
```

```
mRenderer.drawVideoBackground();
```

```
GLS20.glEnable(GLS20.GL_DEPTH_TEST);
```

```
// handle face culling, we need to detect if we are using reflection
```

```
// to determine the direction of the culling
```

```
GLS20.glEnable(GLS20.GL_CULL_FACE);
```

```
GLS20.glCullFace(GLS20.GL_BACK);
```

```
if (Renderer.getInstance().getVideoBackgroundConfig().getReflection() ==  
VIDEO_BACKGROUND_REFLECTION.VIDEO_BACKGROUND_REFLECTION_ON)
```

```
    GLS20.glFrontFace(GLS20.GL_CW); // Front camera
```

```
else GLS20.glFrontFace(GLS20.GL_CCW); // Back camera
```

```
// did we find any trackables this frame
```

```
// [...] traitement (c.f. slide suivant)
```

```
GLS20.glDisable(GLS20.GL_DEPTH_TEST);
```

```
mRenderer.end();
```

- Récupérer l'état de la détection
- Dessiner l'image prise par la caméra

- Différents réglages OpenGL

- Boucle de traitement

- fin...

Traitement d'un résultat (renderFrame)

```
// did we find any trackables this frame
for (int tIdx = 0; tIdx < state.getNumTrackableResults(); tIdx++)
{
    TrackableResult result = state.getTrackableResult(tIdx);

    Trackable trackable = result.getTrackable();
    Matrix44F modelViewMatrix_Vuforia = Tool.convertPose2GLMatrix(result.getPose());

    float[] modelViewMatrix = modelViewMatrix_Vuforia.getData();

    int textureIndex = trackable.getName().equalsIgnoreCase("stones") ? 0 : 1;
    textureIndex = trackable.getName().equalsIgnoreCase("tarmac") ? 2 : textureIndex;

    // deal with the modelview and projection matrices
    float[] modelViewProjection = new float[16];

    if (!mActivity.isExtendedTrackingActive())    {
        Matrix.translateM(modelViewMatrix, 0, 0.0f, 0.0f, OBJECT_SCALE_FLOAT);
        Matrix.scaleM(modelViewMatrix, 0, OBJECT_SCALE_FLOAT, OBJECT_SCALE_FLOAT,
OBJECT_SCALE_FLOAT);
    } else    {
        Matrix.rotateM(modelViewMatrix, 0, 90.0f, 1.0f, 0, 0);
        Matrix.scaleM(modelViewMatrix, 0, kBuildingScale,    kBuildingScale, kBuildingScale);
    }

    Matrix.multiplyMM(modelViewProjection, 0,
vuforiaAppSession.getProjectionMatrix().getData(),
0, modelViewMatrix, 0);

    // [...] // dessin de la théière
}
```

- Récupération des résultats de la détection
- Obtention des matrices de données
 - Changement de repères
 - Mise à l'échelle...
- Dessin...

Du modèle 3D à la position à l'écran

```

Vec3F center = new Vec3F(0,0,0);
Vec2F screenCenter = com.qualcomm.vuforia.Tool.projectPoint(CameraDevice.getInstance().getCameraCalibration(), result.getPose(), center);
float[] screenCenterData = screenCenter.getData();

VideoMode videoMode = CameraDevice.getInstance().getVideoMode(CameraDevice.MODE.MODE_DEFAULT);
VideoBackgroundConfig config = mRenderer.getInstance().getVideoBackgroundConfig();

// on peut le faire une fois et une seule
WindowManager wm = (WindowManager) mActivity.getSystemService(Context.WINDOW_SERVICE);
Display display = wm.getDefaultDisplay();
Point screenSize = new Point();
display.getSize(screenSize); // pour récupérer la taille de l'écran...

int[] configSize = config.getSize().getData();
int[] configPosition = config.getPosition().getData();

// décalage par rapport à l'écran
float xOffset = (screenSize.x - configSize[0]) / 2f + configPosition[0];
float yOffset = (screenSize.y - configSize[1]) / 2f - configPosition[1];

// calcul selon l'orientation, inversion on non des deux axes
Vec2F onScreen ;

if (mActivity.getResources().getConfiguration().orientation == Configuration.ORIENTATION_PORTRAIT) {
    float rotatedX = videoMode.getHeight() - screenCenterData[1];
    float rotatedY = screenCenterData[0];
    onScreen= new Vec2F(rotatedX * ((float) configSize[0]) / (float) videoMode.getHeight() + xOffset,
        rotatedY * configSize[1] / (float) videoMode.getWidth() + yOffset);
} else {
    onScreen = new Vec2F(screenCenterData[0] * configSize[0] / (float) videoMode.getWidth() + xOffset,
        screenCenterData[1] * configSize[1] / (float) videoMode.getHeight() + yOffset);
}

// onScreen contient le x et y, coordonnées à l'écran du centre de l'image reconnue

```

Utiliser Vuforia (*sans image*)

- **Pour faire de la virtualité augmentée**
- Initialisation (possible dans une AsyncTask)

```
Vuforia.setInitParameters(this, Vuforia.GL_20);  
int mProgressValue;  
do { mProgressValue = Vuforia.init(); }  
while (mProgressValue >= 0 && mProgressValue < 100);
```

- Initialisation du tracker

```
TrackerManager tManager = TrackerManager.getInstance();  
ImageTracker tracker = (ObjectTracker)  
    tManager.initTracker(ObjectTracker.getClassType());
```

- Chargement des données

```
mCurrentDataset = tracker.createDataSet();  
mCurrentDataset.load( "sophiatech.xml",  
    STORAGE_TYPE.STORAGE_APPRESOURCE);  
tracker.activateDataSet(mCurrentDataset);
```

Utiliser Vuforia (sans image)

- Détecter les images partielles

```
int numTrackables = mCurrentDataset.getNumTrackables();
for (int count = 0; count < numTrackables; count++) {
    Trackable trackable = mCurrentDataset.getTrackable(count);
    trackable.startExtendedTracking();
    String name = "Current Dataset : " + trackable.getName();
    trackable.setUserData(name);
}
```

- Pouvoir traiter les images (obtenir l'objet State) => cas particulier car sans image

```
mRenderer = new NoRenderer(this);
```

- Initialisation de la caméra (simplifiée car sans image)

```
CameraDevice.getInstance().init(CameraDevice.CAMERA.CAMERA_BACK);
// ou CAMERA.CAMERA_FRONT
CameraDevice.getInstance().selectVideoMode(CameraDevice.MODE.MODE_DEFAULT);
CameraDevice.getInstance().start();
Vuforia.setFrameFormat(PIXEL_FORMAT.RGB565, true);
```

Utiliser Vuforia (sans image)

- Démarrer le tracker

```
TrackerManager tman = TrackerManager.getInstance();  
finalTracker = tman.getTracker(ImageTracker.getClassType());  
finalTracker.start();
```

- Lancer un thread (ou autre) pour suivre la détection

Utiliser Vuforia (sans image)

- Besoin d'un thread pour traiter la détection
- Ici une AsyncTask... mais pas forcément la bonne solution
 - Publication dans le thread graphique..
 - Mais une seule AsyncTask à la fois

```
task = new FollowImage();  
task.execute();
```

- Penser à onPause / onDestroy / onResume
 - C.f. exemple

Utiliser Vuforia (sans image)

- doInBackground

```
protected Void doInBackground(Void... voids) {
    Vec2F center = null;
    Vec2F oldCenter = null;
    while (active) {
        if (mRenderer != null) {
            oldCenter = center;
            center = mRenderer.detect(); // le NoRenderer
            if (center != null) publishProgress(center, oldCenter);
            // s'il y a deux valeurs, on peut faire un delta...
            // ... et faire en fonction du déplacement du centre
            // ou tout autre langage d'interaction...
        }
        SystemClock.sleep(10);
    }

    return null;
}
```

Utiliser Vuforia (sans image)

- Méthode detect()

```
public Vec2F detect() {  
    // récupération du Renderer interne à Vuforia...  
    Renderer mRenderer = Renderer.getInstance();  
    // ... pour obtenir l'état de la détection  
    State state = mRenderer.begin();  
  
    // pour calculer (ou non) la position du centre de  
    // la 1er image détectée  
    Vec2F onScreen = null;  
  
    // traitement d'une seule détection  
    if (state.getNumTrackableResults() > 0) {  
        // [...] // calcul de onScreen, c.f. slide 28  
    }  
    mRenderer.end();  
    return onScreen;  
}
```

Exploitation(s)...

- De nouveaux dispositifs au sens de la **multimodalité**
- Selon la caméra, les gestes changent (inverses)
- **En TP : à vous de proposer des modalités à partir de vuforia**
 - Le plus simple : déplacement du centre de déformation...

Références (non insérées)

- [Milgram] Milgram, Kishino. A taxonomy of Mixed Reality Visual Displays. IEICE Transactions on Information Systems, vol. E77-D, n°12, December 1994, 11 pages.
- [Feiner] Feiner, MacIntyre, Höllerer, Webster. A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. in Proceedings of the International Symposium on Wearable Computers (ISWC '97), Cambridge, October 1997, pp 74-81.
- [Human Pacman] Cheok, Yang, Ying, Billingham, Kato. Touch-Space: Mixed Reality Game Space Based on Ubiquitous, Tangible, and Social Computing. Personal and Ubiquitous Computing, Vol. 6, No 5/6, London, 2002, pp 430-442.
- [ARQuake] Thomas, Close, Donoghue, Squires, De Bondi, Morris, Piekarski. ARQuake: An Outdoor/Indoor Augmented Reality First Person Application. In Proceedings of Fourth International Symposium on Wearable Computers (ISWC 2000), IEEE Computer Society, Atlanta, October 2000, pp. 139-146.
- [Kraut] Kraut, Miller, Siegel. Collaboration in Performance of Physical Tasks: Effects on Outcomes and Communication. In Proceeding of CSCW'96, 1996. pp. 57-66.
- [Mah-jong] Kortuem, Segall, Bauer. Context-Aware, Adaptive Wearable Computers as Remote Interfaces to 'Intelligent' Environments. 2ND. International Symposium on Wearable Computers, Pittsburgh, October 1998, pp. 58- 65.
- [Can you see me now ?] <http://www.canyouseemenow.co.uk/> et <http://www.mrl.nott.ac.uk/~axc/documents/>
- [Virtual Showcase] Bimber, Gatesy, Witmer, Raskar, Encarnação. Merging Fossil Specimens with Computer-Generated Information. In IEEE Computer, vol 35, n° 9, September 2002, Electronic Edition (IEEE Computer Society DL), pp 25-30.