# Soutenance de Projet Android

Arthur Wenger & Shad Maleck

M1 Informatique Université de la Réunion

13 novembre 2017

# Introduction

## Objectif

- Créer un jeu simple sur Android

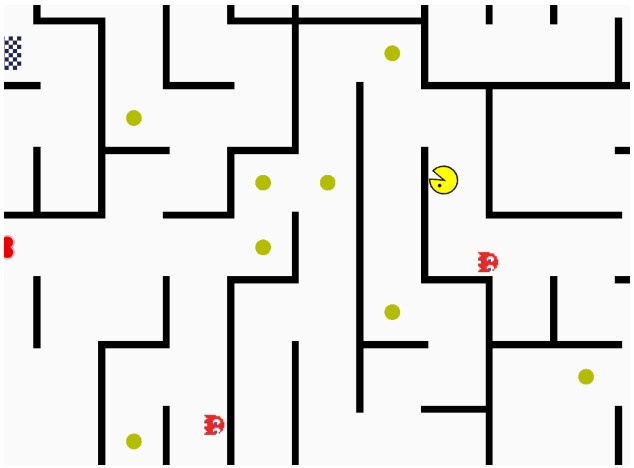
## Contraintes

- Utiliser un capteur pour déplacer une balle
- Créer une liste de scores
- Gérer la persistance des scores
- Créer une carte pour visualiser les scores
- Ajouter du son

# Présentation du jeu

## Principe

- Reprendre le concept de Pacman

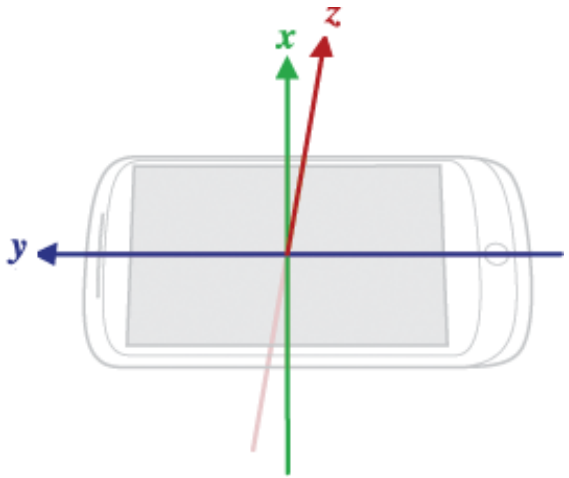


## Implications

- Créer un labyrinthe
- Utiliser accéléromètre pour déplacer pacman
- Gérer les collisions avec les murs et les objets du jeu

# L'accéleromètre

- Mesure l'accélération de l'appareil selon 3 axes :




$$\vec{a_d} = -\frac{\sum \vec{F}}{m} - \vec{g} \text{ avec } \|\vec{g}\| = 9.81\,\mathrm{m\,s^{-2}}$$

- Quand l'appareil ne bouge pas : $\sum \vec{F} = 0 \implies \vec{a_d} = -\vec{g}$
- L'accélération détermine l'orientation du téléphone
- Mais l'accélération est aussi une variation de vitesse :

$$\vec{a} = \frac{d\vec{v}}{dt} = \frac{d^2\vec{x}}{dt^2}$$

# Les mouvements de Pacman

- Capture et transmission de l'accélération :

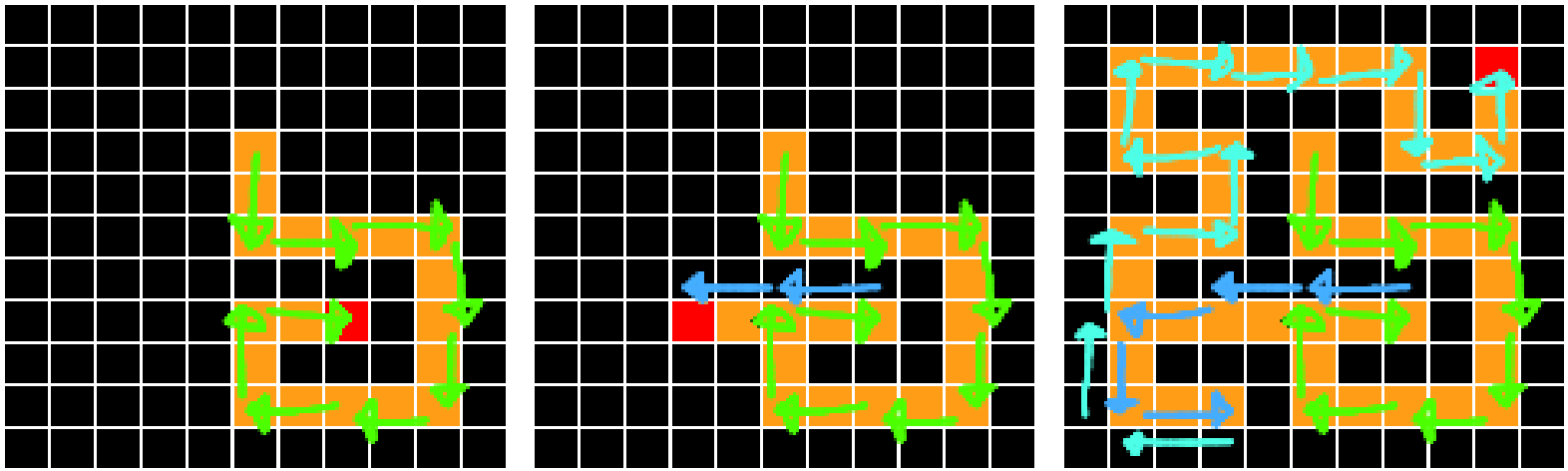
```
manager.registerListener(sensorListener, mAccelerometer
    , SensorManager.SENSOR_DELAY_UI);

private class MySensorListener implements
    SensorEventListener {
  public void onSensorChanged( SensorEvent event ) {
if (event.sensor.getType()==Sensor.TYPE_ACCELEROMETER)
mazeView.updateAccel(-event.values[0],event.values[1]);
}}
```

- Mise à jour de l'accélération de pacman
- Prise en compte de l'accélération en utilisant la formule : $v_f = v_i + a * dt$ avec $dt = \frac{1}{FPS}$

# Le labyrinthe

- Génération aléatoire avec l'algorithme « Depth-first search »



- Création des murs avec une classe modélisant un segment
- Initialisation des positions des objets du jeu

# La vue et les objets du jeu

## La vue

- Une vue spécifique « GameView »
- La méthode onDraw pour dessiner le labyrinthe et les objets
- Un thread pour mettre à jour le jeu

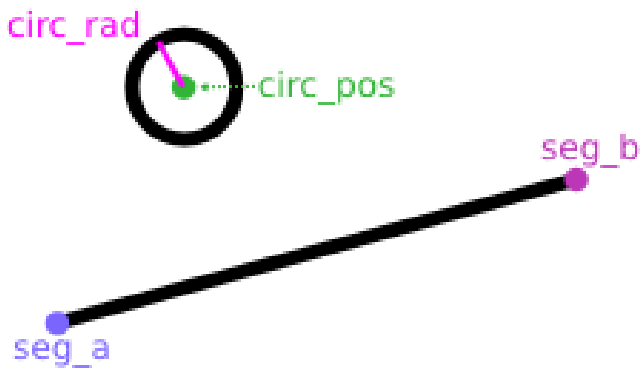
## Les objets

- Des sprites pour représenter pacman et les ennemis

- Une méthode draw pour chaque objet du jeu
- Une méthode de déplacement spécifique pour pacman et les fantômes

# Detection des collisions

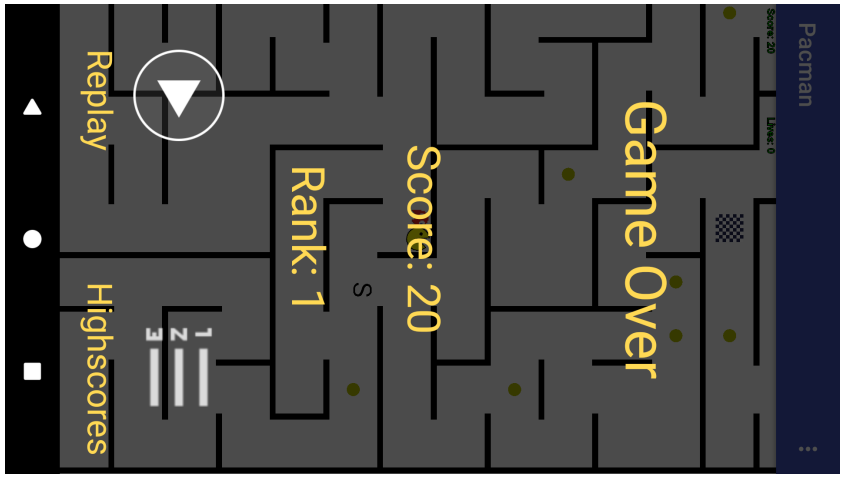
- Chaque objet est associé à une forme géométrique
- Détecter les intersections entre des formes simples




- Ajuster le mouvement des objets

# Les scores

- Le nombre de pièces accumulées détermine le score



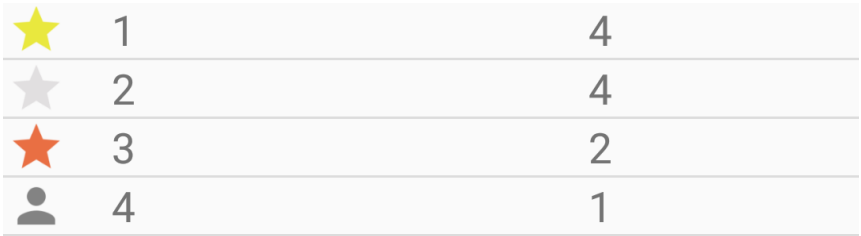

- Une classe spécifique pour modéliser les scores :

```
public class Score implements Serializable {
private int rank, value;
private double lat, lng; ...
```

- Une activité pour l'affichage d'une liste de scores
- Gestion de la persistance avec une base de données SQLite

# Liste de scores

- La liste est accessible à de multiples endroits : menu du jeu, menu principal, ecran de game over
- La vue « ListView » pour afficher une liste




- Un « ArrayAdapter » pour réaliser l'interface entre les scores et la vue
- La méthode getView pour afficher un score à une position spécifique dans la liste

```
Score score = getItem(position);
int rank = score.getRank();
int value = score.getValue();
rankView.setText( String.valueOf(rank) );
scoreView.setText( String.valueOf( value ) );
switch(rank) { /* selection de l'image en fonction du
    rang... */ }
return cellView;
```

# Persistance

- Conservation des scores avec une base de données SQLite :

```
public class DBManager extends SQLiteOpenHelper {
  ...
  public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table scores (" +
      SCORES_COLUMN_ID+" integer primary key
          autoincrement, " +
      SCORES_COLUMN_VALUE+" integer, " +
      SCORES_COLUMN_LATITUDE+" real, " +
      SCORES_COLUMN_LONGITUDE+" real)"
  );}
```

- « getWritableDatabase » et « getReadableDatabase » créent, ouvrent ou mettent à jour la base
- Un fichier « ScoresDB.db » est crée dans un dossier protégé
- Permet d'interroger la base avec des requêtes SQL simples

# Localisation du téléphone

- Récupérer une seule fois la position
- Utiliser la méthode « requestSingleUpdate » de la classe « LocationManager »

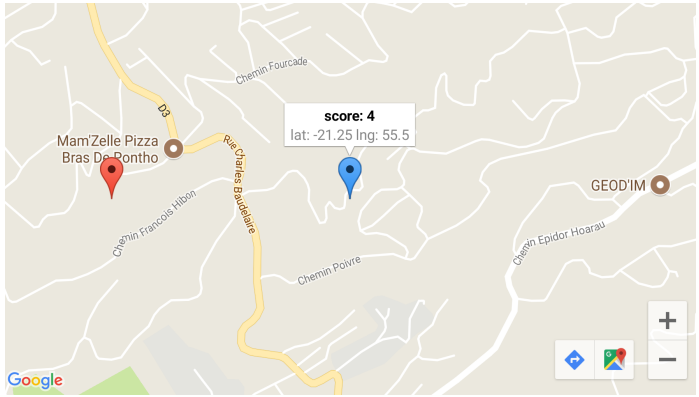
```
locationManager.requestSingleUpdate( criteria, new
    MyLocationListener( callback ), null );

private static class MyLocationListener implements
    LocationListener { ...
  public void onLocationChanged( Location location ) {
   callback.onNewLocationAvailable( new Double[] {
       location.getLatitude(), location.getLongitude()}
   );}
```

- Prendre en compte les autorisations et la latence
- Transmission des coordonnées avec la méthode « putExtra » de la classe « Intent »

# La carte

- Utilisation de l'API Google Map
- Un simple fragment pour le layout :




- Récupération des coordonnées des scores
- « onMapReady » pour l'affichage des marqueurs de position

```
int centerScoreRank = centerScore.getRank();
for(Score score: scores_array){
    mMap.addMarker( createMarker( score, score.getRank
        ()==centerScoreRank ) );
}
LatLng centerLatLng = new LatLng( centerScore.
    getLatitude(), centerScore.getLongitude() );
mMap.animateCamera( CameraUpdateFactory.newLatLngZoom(
    centerLatLng, 11 ) );
```

# Les sons

- Utilisation de la classe MediaPlayer
- Lecture d'un fichier du dossier « Assets »

```
player = new MediaPlayer();
player.setVolume(volume, volume);
player.setDataSource(afd.getFileDescriptor(), afd.
    getStartOffset(), afd.getLength());
player.setLooping(loop);
player.prepare();
player.setOnCompletionListener( new MediaPlayer.
    OnCompletionListener() {
  public void onCompletion( MediaPlayer mp ) {
    stopPlayer();
  }});
```

- Gestion de la libération des ressources

# Conclusion

- Création d'un jeu simple intégrant un modèle physique
- Un ensemble d'activités et d'objets qui coopèrent
- Utilisation des fonctionnalités du téléphone : capteurs, services...
- Un jeu perfectible