Rapport CCTP Développement Mobile

Grégory GUICHARD, L3 informatique

14 avril 2019

Résumé

Durant l'année de L3 informatique, les étudiants participent à une UE de développement mobile. Ils doivent programmer un jeu sur Android et iOS.

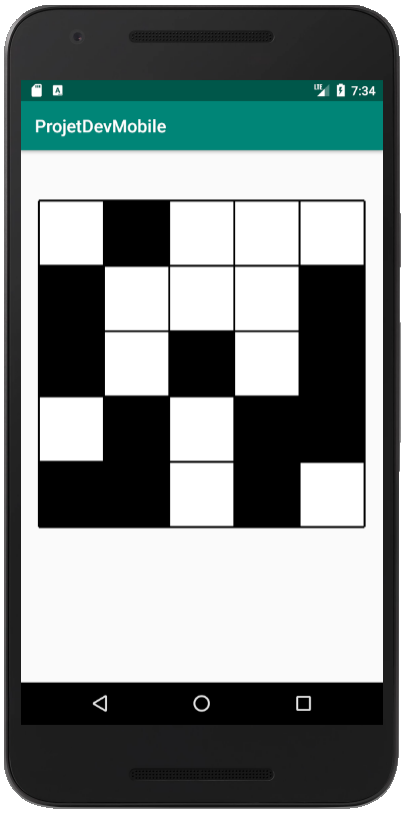
1 INTRODUCTION

Pour ce TP, il faut concevoir un jeu pour Android[1] et iOS[4] en respectant quelques contraintes :

- Un score doit être associé à chaque partie
- L'application doit posséder plusieurs écrans
  - Un écran d'acceuil
  - Un écran pour pourvoir enregistrer son score à la fin de la partie
  - Un écran pour consulter les scores
- Le joueur doit avoir la possibilité d'enregistrer et de supprimer son score de façon permanente

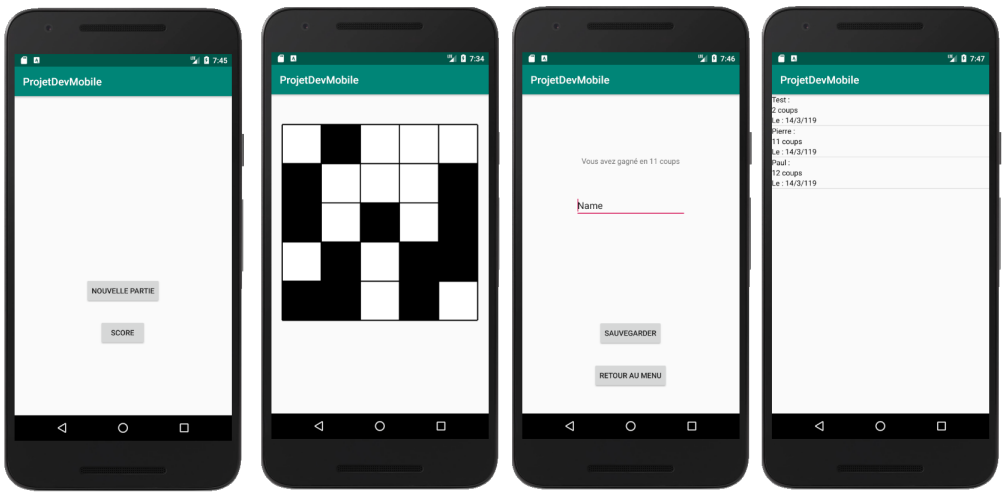
2 Le jeu

Le joueur dispose d'une grille de 25 cases blanches réparties sur 5 lignes. Le but du jeu est de colorer toutes les cases en noir. Pour se faire, le joueur clique sur la case qu'il veut colorer mais il colore également les cases adjacentes (celle du haut, du bas, à gauche et à droite). Si la case à colorer est blanche, elle devient noire et inversement.



3 Interface

L'application possède 4 écrans pour Android et 3 écrans pour iOS
— Un écran d'acceuil permettant de commencer une nouvelle partie ou de voir les scores,
— Un écran où le joueur joue,
— Un écran permettant d'afficher et d'enregistrer son score à la fin d'une partie,
— Un écran permettant d'afficher tous les scores enregistrés.



4 La grille

La grille est déssinée automatiquement par le code. Cela permet de dessiner des grilles de tailles différentes simplement en modifiant les nombre de colonnes et de lignes. On dessine d'abord le fond des cellules puis les lignes qui forment la grille.

La fonction onDraw[2] pour dessiner la grille sur Android.

```
@Override
    protected void onDraw(Canvas canvas) {

        // --- On dessine les cellules ---
        for( int y=0; y<5; y++ ) {
            for( int x=0; x<5; x++ ) {

                int backgroundColor = Color.WHITE;

                // On change la couleur du fond si la valeur est à 1
                if ( gameBoard.cells[y][x].Value==1 ) {
                    backgroundColor = Color.BLACK;
                }

                // Draw the background for the current cell
                paint.setColor( backgroundColor );
                canvas.drawRect(x * cellWidth+marginGrid,y * cellWidth + 150 ,
                    (x+1) * cellWidth+marginGrid,(y+1) * cellWidth + 150, paint);
            }
        }

        // --- On dessine les lignes de la grille ---
        paint.setColor( Color.BLACK );
        paint.setStrokeWidth( gridSeparatorSize );
        for( int i=0; i<=5; i++ ) {
            canvas.drawLine( i*cellWidth+marginGrid, 150,
                i*cellWidth+marginGrid, 150+cellWidth*5, paint );
            canvas.drawLine( marginGrid,150+i*cellWidth,
                cellWidth*5+marginGrid, 150+i*cellWidth, paint );
        }
    }
```

La fonction onDraw() pour dessiner la grille sur iOS

```
    override func draw(_ rect: CGRect) {

        let context = UIGraphicsGetCurrentContext()

        let w = Int(rect.size.width)
        let cellwidth = w/5
        //Dessiner les cellules

        for x in 0..<5 {
            for y in 0..<5 {
                if (gb.getGameBoard()[x][y].value==0){
                    context?.setFillColor(red: 0, green: 0, blue: 0, alpha: 0.5)
                } else {
                    context?.setFillColor(red: 0, green: 0, blue: 0, alpha: 100)
```

```
                }
                let rectangle = CGRect(x: x*cellwidth, y: y*cellwidth,
                    width: cellwidth, height: cellwidth)
                context?.fill(rectangle)
            }
        }

        //Dessiner les lignes
        context?.setLineWidth(2)
        UIColor.black.setStroke()

        for i in 0..<6 {
            context?.move(to: CGPoint(x: i*cellwidth, y: 0))
            context?.addLine(to: CGPoint(x: i*cellwidth, y: cellwidth*5))
            context!.strokePath()

            context?.move(to: CGPoint(x: 0, y: i*cellwidth))
            context?.addLine(to: CGPoint(x: cellwidth*5, y: i*cellwidth))
            context!.strokePath()
        }
    }
```

5 La gestion des évènements

Lors de la partie, il suffit de toucher l'écran sur grille pour pourvoir colorer les cases. La gestion de cet évènement s'effectue grâce à :

— gestureDetector et à sa fonction onSingleTapUp() pour Android
— UITouch et à sa fonction touchesBegan() pour iOS

6 La persistance des données

La persistance des données n'est présente que sur Android. Elle est réalisé grâce à une sauvegarde dans une base de données relationnelle. La base de données possède une seule table "tableScore" composée de 4 champs :

— "idScore" pour reconnaitre chaque donnée
— "nom" pour le nom du joueur
— "score" pour le score
— "date" pour la date à laquelle le score a été réalisé

La création et la modification de la base de données s'effectuent grâce à "SQLiteDataBase"[3]

7 Conclusion

Pour ce TP, le programme sous Android fonctionne, le joueur peut lancer une partie, sauvegarder son score si il le souhaite, afficher les scores enregistrés et effacer les scores souhaités. Sous iOS, le joueur peut lancer une nouvelle partie et la terminer. Mais lors de l'affichage des scores, la variable n'est pas transmise à l'écran suivante. J'ai essayé de mettre en place une solution grâce à une fonction Delegate mais je n'ai pas réussi.

Références

[1] Android. https://developer.android.com/docs.

[2] Dessiner la grille(android). http://koor.fr/Java/Android/CurveTracer.wp.

[3] Sqlite. http://koor.fr/Java/Android/SQLiteSample.wp.

[4] Swift. https://swift.org/documentation/.