Core data & Capteurs : Création d'un jeu.

IMT Nord Europe

De nombreux jeux, sur iOS et Android, utilisent les capteurs. Ainsi, on trouve des imitations des jeux de fêtes foraine avec le coup de marteau et la mesure de force, ou encore tout autre mesure d'activité ou d'action que l'on pourrait imaginer. Vous allez faire une application comme celle-ci, utilisant les capteurs et une base de données pour sauvegarder des joueurs et leurs scores. À noter également que si vous ne vouliez, dans votre jeu, n'enregistrer que les dix meilleurs scores par exemple, utiliser une base de données n'est pas obligatoire. Vous pourriez utiliser les NSUserDefault, dans lequel vous avez d'une part les préférences comme la semaine dernière, mais aussi toute donnée de taille faible que vous voudrez enregistrer dans votre application. L'utilisation de base de données ici vous est montré pour vous en apprendre le minimum sur cette partie importante de la programmation iOS.

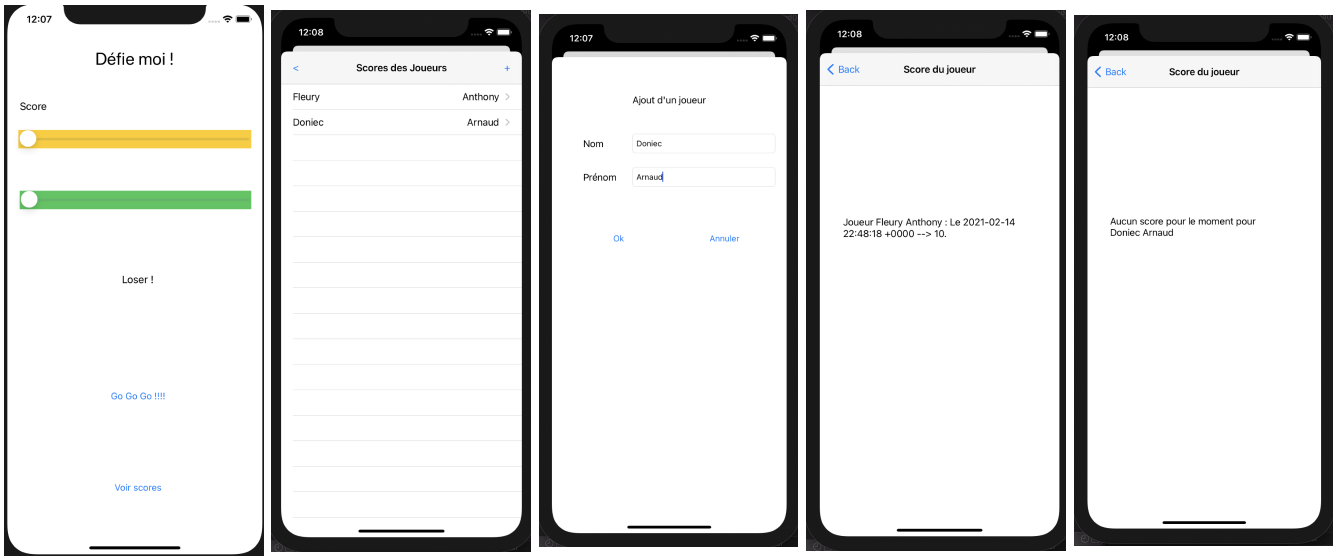

FIGURE 1 – Exemple de base d'application réalisée à partir de ce sujet.

1 Création du projet

Vous allez cette fois créer un projet dans lequel vous préciserez, en cochant la case correspondante, que vous voulez utiliser Core Data.
Cette application va contenir une fenêtre unique, un storyboard ainsi qu'un modèle de données. Nous allons construire au fur et à mesure des fenêtres supplémentaires.

2 Core data : Changement du modèle de données

Entrez dans le fichier xcdatamodeld qui montre le modèle de données de votre application.
Pour l'exemple, ici, nous avons créé une entité qui rassemble les informations liées à la personne (son nom et son prénom) et une entité qui rassemble les différents scores de ce joueur et qui est liée à celle possédant le nom et le prénom. La figure 2 vous montre le modèle créé. Pour ce modèle, nous créons d'abord des entités, avec les différentes données contenues et leur type, puis les relations. Vous pouvez aussi régler le fait que chacun des champs soit optionnel ou non, qu'il soit indexé, etc.
Créez alors le modèle de données montré sur la figure avec ses entités et ses relations.
Pour les relations, vous avez une relation qui part de chacune des entités, et la relation inverse de chacune est l'autre relation (afin que vous ayez le même affichage que celui présenté). La relation qui va du joueur au score est de type "To many". Veillez à bien changer cela.
Ensuite, vous allez créer des classes pour ce modèle. Pour cela, sélectionnez vos deux entités. Dans le menu Editor, cliquez sur Create NSManagedObject subclass. Cela va vous créer une classe pour chacune des entités, que vous pourrez utiliser par la suite.

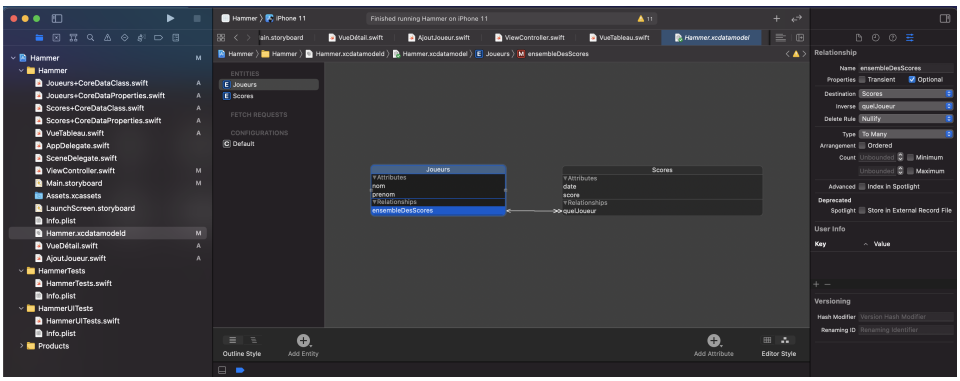

FIGURE 2 – Modèle de données

Note importante, dans tous les fichiers swift dans lesquels vous utiliserez CoreData, vous mettrez en haut du fichier import CoreData

3 Préparation de la fenêtre de jeu

Ajoutez différents contrôles pour arriver à une fenêtre telle que celle proposée plus haut (avec deux sliders, deux boutons, un peu de texte etc.). Les codes suivants vont vous permettre de faire quelques manipulations sur cette fenêtre :

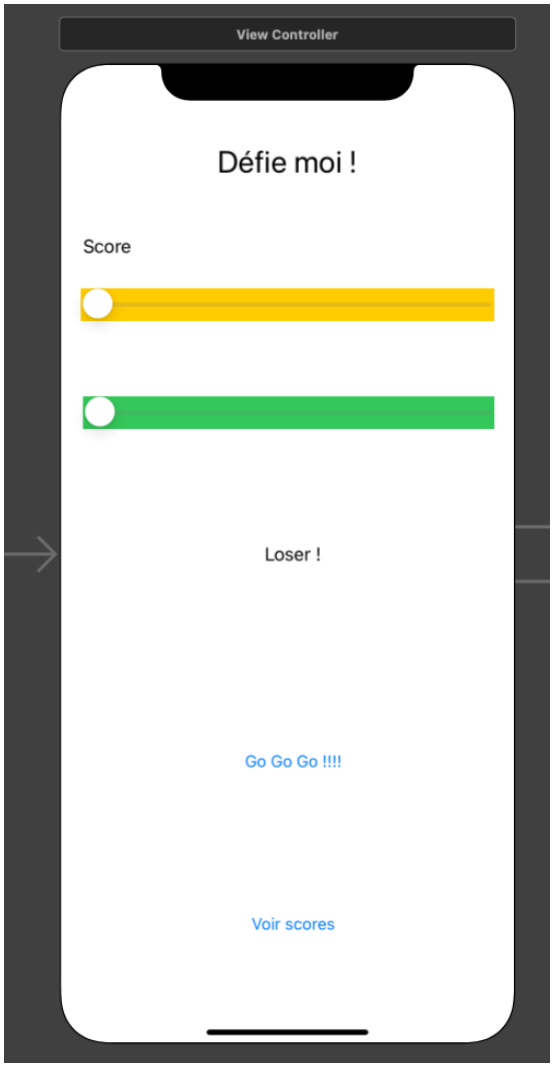

FIGURE 3 – Fenêtre de jeu.

3.1 Variables utiles

```
var motionManager: CMMotionManager!
var timer: Timer!
var nbVal: Int = 0
var donnees: [Double] = []
var score: Double = 0.0
var score2: Double = 0.0
var joueurAAssigner: Joueurs?

@IBOutlet var message: UILabel?
@IBOutlet var boutonGo: UIButton?
@IBOutlet var boutonScores: UIButton?
@IBOutlet var slider1: UISlider?
@IBOutlet var slider2: UISlider?
```

3.2 Méthodes du bouton de jeu

```
@IBAction func GoGoGo(_ sender: UIButton) {
    nbVal = 0
    motionManager = CMMotionManager()
    motionManager.accelerometerUpdateInterval = 0.01
    motionManager.startAccelerometerUpdates()
    donnees = [Double](repeating: 0, count: 500)

    // ici : décoration pour faire une jolie animation avec des chiffres
    // possiblement parler : classe AVSpeechVoiceSynthesis : https://developer.apple.com
        /documentation/avfaudio/avspeechsynthesisvoice

    boutonGo?.isEnabled = false
    boutonScores?.isEnabled = false
    timer = Timer.scheduledTimer(timeInterval: 0.01, target: self, selector: #selector(
        mesureDonnees), userInfo: nil, repeats: true)
}
```

3.3 Mesure et score

```
@objc func mesureDonnees(timer: Timer) {
    if(nbVal >= 500) {
        motionManager.stopAccelerometerUpdates()
        timer.invalidate()
        motionManager = nil
        self.calculScore()
        boutonGo?.isEnabled = true
        boutonScores?.isEnabled = true
        self.performSegue(withIdentifier: "segueSelectionJoueur", sender: self)
    }
    else {
        if(motionManager == nil) {
            return
        }

        //donnees[nbVal] = sqrt(motionManager.accelerometerData!.acceleration.x *
            motionManager.accelerometerData!.acceleration.x + motionManager.
            accelerometerData!.acceleration.y * motionManager.accelerometerData!.
            acceleration.y + motionManager.accelerometerData!.acceleration.z *
            motionManager.accelerometerData!.acceleration.z)

        //attention sur simulateur, accelerometerData est nil...
        print("Acquisition : \(nbVal)")
        nbVal += 1
    }
}
```

```
    func calculScore() {
        // Calcul de votre score en fonction du max ou de la moyenne de votre tableau "
            donnees"
        // Affectation dans la variable score...

        score = 10
    }
```

Ce calcul sur les données est un exemple pour vous montrer où sont ces données, à vous de faire ce que bon vous semble.

4 Vue tableau pour les scores et les joueurs et les détails

Nous allons maintenant voir comment faire, avec CoreData, un affichage des joueurs et de leurs scores.

4.1 Préparation de votre storyboard et de votre projet

Vous allez pour cela :

- Ajouter un contrôle de type "Navigation Controller"
- Ajouter une classe de type "Cocoa Touch Class" qui dérive de UITableViewController
- Associer la classe au Root View Controller de votre navigation
- Ajouter une vue supplémentaire à côté pour le détail de vos scores
- Ajouter une classe de type "Cocoa Touch Class" qui dérive de UIView
- Associer la classe et la nouvelle vue

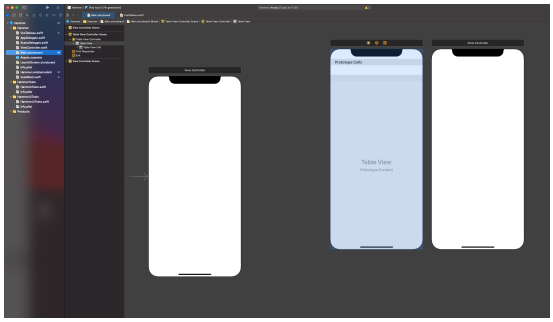

FIGURE 4 – Ajout du contrôleur de navigation.

4.2 La classe associée à votre controlleur de vue tableau

Avant tout, en plus du UITableViewController, vous allez ajouter , NSFetchedResultsControllerDelegate comme protocole supporté.
Dans cette classe, nous allons "charger" les données. Pour cela il vous faut des variables et méthodes dans votre classe :

```
let persistentContainer = NSPersistentContainer.init(name: "Hammer") // ici à remplacer par
    le nom de votre modèle

lazy var fetchedResultsController: NSFetchedResultsController<Joueurs> = {
    let fetchRequest: NSFetchRequest<Joueurs> = Joueurs.fetchRequest()

    fetchRequest.sortDescriptors = [NSSortDescriptor(key: "nom", ascending: false)]

    let appDelegate = UIApplication.shared.delegate as! AppDelegate
    let managerContext = appDelegate.persistentContainer.viewContext

    let fetchedResultsController = NSFetchedResultsController(fetchRequest: fetchRequest
        , managedObjectContext: managerContext, sectionNameKeyPath: nil, cacheName:
        nil)
    fetchedResultsController.delegate = self
```

```
        return fetchedResultsController
    }()

    func chargerDonnees(){
        persistentContainer.loadPersistentStores { (persistentStoreDescription, error) in
            if let error = error {
                print("Unable to Load Persistent Store")
                print("\(error), \(error.localizedDescription)")
            } else {
                do {
                    try self.fetchedResultsController.performFetch()
                } catch {
                    let fetchError = error as NSError
                    print("Unable to Perform Fetch Request")
                    print("\(fetchError), \(fetchError.localizedDescription)")
                }

            }
        }
    }
```

Ensuite, ceci sera appelé dans les méthodes suivantes (viewDidLoad est appelée quand votre fenêtre est chargée et vous sert à initialiser des choses, la seconde est appelée chaque fois que votre fenêtre réapparait) :

```
    override func viewDidLoad() {
        super.viewDidLoad()

        chargerDonnees()
        self.tableView.reloadData()
    }

    override func viewDidAppear(_ animated: Bool) {
        self.tableView.reloadData()
    }
```

```
override func numberOfSections(in tableView: UITableView) -> Int {
        return 1
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) ->
        Int {
        guard let jouueurs = fetchedResultsController.fetchedObjects else { return 0 }
        return jouueurs.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
        UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "celluleJeu", for:
            indexPath)
        let joueur = fetchedResultsController.object(at: indexPath) as Joueurs

        cell.textLabel?.text = joueur.nom
        cell.detailTextLabel?.text = joueur.prenom

        return cell
    }

    override func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.
        EditingStyle, forRowAt indexPath: IndexPath) {
        if editingStyle == .delete {
            let managedObjectContext: NSManagedObjectContext = (UIApplication.shared.
                delegate as! AppDelegate).persistentContainer.viewContext
            let managedObject: NSManagedObject = fetchedResultsController.object(at:
                indexPath) as NSManagedObject
                    managedObjectContext.delete(managedObject);
```

```
                    do {
                        try managedObjectContext.save();
                    } catch {
                        // Traitement Erreur...
                    }
        }
    }
```

4.3 Dans storyboard : associer votre vue avec le chargement des scores

Faites en sorte de charger cette vue lorsque l'utilisateur appuie sur le bouton pour voir les scores.

4.4 Bouton retour

Ajoutez un BarButton dans votre fenêtre en haut à gauche et mettez par exemple comme texte "<" pour montrer un retour. Associez celui-ci à la méthode suivante :

```
    @IBAction func retour(_ sender:UIButton) {
        self.dismiss(animated: true, completion: nil)
    }
```

5 Un appel différent de la vue

Ajoutez un second "chargement" de cette vue, à partir de la vue principale du jeu, en mettant manual comme Segue cette fois. Vous appelerez cette Segue "segueSelectionJoueur" pour faire écho au nom donné dans mesureDonnees car cette méthode va "charger" cette fenêtre uniquement pour y sélectionner un joueur à qui attribuer un score.
Pour cela il faudra :

— Ajouter ces variables dans la vue tableau :

```
    var estAppeleeParSelection = false
    var appelant: UIViewController?

    var JoueurEnCours: Joueurs?
```

— Ajouter ce code pour la sélection dans la vue tableau :

```
override func tableView(_ tableView: UITableView,
                        didSelectRowAt indexPath: IndexPath) {
        if(self.estAppeleeParSelection) {
            (self.appelant as! ViewController).joueurAAssigner =
                    fetchedResultsController.object(at: indexPath) as Joueurs
            self.dismiss(animated: true, completion: nil)
            (self.appelant as! ViewController).saveScore()
        }
        else {
            JoueurEnCours = fetchedResultsController.object(at: indexPath) as Joueurs
            self.performSegue(withIdentifier: "segueDetail", sender: self)
        }
    }
```

— Ajouter ce code dans la vue du jeu :

```
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if(segue.identifier == "segueSelectionJoueur") {
            ((segue.destination as! UINavigationController).topViewController as!
                    VueTableau).estAppeleeParSelection = true
            ((segue.destination as! UINavigationController).topViewController as!
                    VueTableau).appelant = self
        }
    }
```

— Vous pouvez alors faire votre sauvegarde de score dans la vue du jeu :

```
    func saveScore() {
        let max1: Double = sqrt(192) // Si 8g max par axe...

        if(score > 10) {
            message?.text = "BIEN"
        }
        else if(score > 5) {
            message?.text = "Pas Mal"
        }
        else {
            message?.text = "Loser !"
        }

        slider1?.value = Float(score / max1)
        slider2?.value = Float(score / max1)

        let appDelegate = UIApplication.shared.delegate as! AppDelegate
        let managerContext = appDelegate.persistentContainer.viewContext
        let s: Scores = NSEntityDescription.insertNewObject(forEntityName: "Scores",
                into: managerContext) as! Scores

        s.score = Int16(score)
        s.date = Date()
        s.quelJoueur = joueurAAssigner

        if(joueurAAssigner?.ensembleDesScores == nil) {
            let setScores = NSSet.init(object: s)
            joueurAAssigner?.ensembleDesScores = setScores
        }
        else {
            joueurAAssigner?.addToEnsembleDesScores(s)
        }
        do {
            try managerContext.save()
            print("Ajout ok")
        } catch {
            let fetchError = error as NSError
            print("Impossible d'ajouter")
            print("\(fetchError), \(fetchError.localizedDescription)")
        }
    }
```

6 La vue détails

La seconde vue que vous avez créée et à laquelle vous avez associé une classe vous servira à afficher les détails des scores d'un joueur. Cette vue va être associée à votre vue tableau par une segue nommée "segueDetail" pour faire référence au nom donné dans la sélection d'une ligne de la vue tableau.
Elle contiendra le code suivant :

```
    var JoueurEnCours: Joueurs?
    @IBOutlet var textAAfficher: UILabel?
    var texte:String = ""

    override func viewDidLoad() {
        super.viewDidLoad()
        textAAfficher?.text = texte
    }

    func afficherJoueur() {
        if(JoueurEnCours == nil) {
            return
        }
        if(JoueurEnCours?.ensembleDesScores == nil || JoueurEnCours?.ensembleDesScores?.
                count == 0) {
```

```
            let j:Joueurs = JoueurEnCours!
            texte = "Aucun score pour le moment pour \(j.nom!) \(j.prenom!)"
        }
        else {
            let j:Joueurs = JoueurEnCours!
            let ensembleScores:NSArray = j.ensembleDesScores!.allObjects as NSArray
            texte = "Joueur \(j.nom!) \(j.prenom!) : "
            for index in 0..<ensembleScores.count-1 {
                let s = ensembleScores.object(at: index) as! Scores
                texte += "Le \(s.date!) -> \(s.score) / "
            }
            let s = ensembleScores.object(at: ensembleScores.count-1) as! Scores
            texte += "Le \(s.date!) -> \(s.score)."
        }
        print("ok")
    }
```

La variable de type UILabel sera associée au label de la fenêtre et permettra de remplir le texte.

7 Ajout d'un joueur

Vous allez maintenant ajouter une dernière fenêtre, qui va se présenter comme sur la figure tout au début, et qui permettra d'ajouter un joueur. Ajoutez la fenêtre dans le storyboard et une classe dérivant de UIViewController dans le projet que vous associez à cette fenêtre. Associez également un autre BarButton dans votre vue tableau, cette fois ci avec un "+" qui sera associé à l'ouverture de cette fenêtre. Votre storyboard au final devra ressembler à l'image suivante.

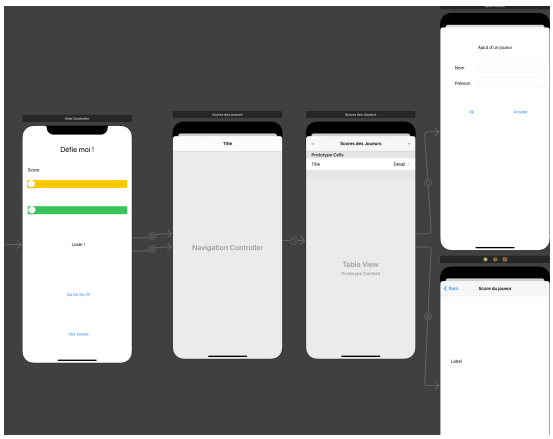

FIGURE 5 – Storyboard final.

Si l'on considère que vous avez deux zones de texte, une pour le nom associée à une variable n et une pour le prénom associée à une variable p, vous aurez le code suivant :

```
    @IBOutlet var n:UITextField?
    @IBOutlet var p:UITextField?

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func ok(_ sender: UIButton) {
        let appDelegate = UIApplication.shared.delegate as! AppDelegate
        let managerContext = appDelegate.persistentContainer.viewContext
        let j: Joueurs = NSEntityDescription.insertNewObject(forEntityName: "Joueurs", into:
                managerContext) as! Joueurs
        j.nom = n?.text
        j.prenom = p?.text
        j.ensembleDesScores = nil

        do {
```

```
            try managerContext.save()
            print("Ajout ok")
        } catch {
            let fetchError = error as NSError
            print("Impossible d'ajouter")
            print("\(fetchError), \(fetchError.localizedDescription)")
        }
        self.dismiss(animated: true, completion: nil)
    }

    @IBAction func cancel(_ sender: UIButton) {
        self.dismiss(animated: true, completion: nil)
    }
```

7.1 Test de votre application

Vous avez maintenant un jeu lorsque vous aurez implémenté le calcul de score. Testez ceci.

8 À vous de jouer... Vous avez carte blanche.

Vous avez, avec cette première partie, la base pour faire un jeu basé sur Core Data. Maintenant à vous de jouer. Faites, pour le rendu, la meilleure application possible permettant mesurer différents joueurs sur le mouvement ou l'activité que vous voulez.
Vous êtes encouragés à faire une application « jolie »avec aux choix, icônes, fonds d'écran, règles de jeu que vous déterminerez, déroulement du jeu que vous imaginerez, ajouts tels qu'un bouton permettant d'avoir le top des scores entre tous les joueurs, etc. Vous serez noté sur l'ajout que vous pourrez apporter à l'application présentée dans ce TP.