

COLLÈGE AHUNTSIC

AEC-420-290

Laboratoire 1

OBJECTIF

Le but de ce laboratoire est de mettre en pratique les notions basiques de Swift. Pour ce faire nous allons créer un programme pour le terminal pour jouer au jeu « Deviner le nombre ».

CRITERE D'ÉVALUATION

Le laboratoire est noté selon le nombre de fonctionnalités implémentées et la qualité du code.

Vous devez remettre :

- Un fichier zip qui contient votre projet 02
- et le nommer NNNp_201234567_labo02.zip
 - 'NNN' sont les trois premières lettres de votre nom de famille
 - 'p' est la première lettre de votre prénom
 - 201234567 est votre matricule
- Une présentation doit être faite devant la classe le cours suivant la remise. Si un étudiant ou un groupe ne présente pas une pénalité de 50% sera appliquée.

PÉNALITÉ EN CAS DE RETARD

Les pénalités sont comptées à la minute à titre de $\frac{1.25}{60}\%$ de votre total de points par minute, ou encore -1.25%/heure, ou encore -30%/24h. Par exemple, un travail remis à 15h30 à la place de 12h00, la journée de remise va être pénalisé de -4.375% du total de points que l'étudiant aurait reçu normalement.

RESSOURCES

- 1) Doc de Swift : [lien](#)

TRAVAIL DEMANDÉ

Soit le jeu non fonctionnel suivant :

```
//
//  main.swift
//  Created by Antoine Moevus

import Foundation

var str = "Bienvenue au jeu!"
let borneMin = 1
let borneMax = 100
let chiffreMystère = Int(arc4random_uniform(10) + 1) // Écrire la formule qui
permet de calculer un chiffre entre borneMin et borneMax
let nbCoupsMax = 3

print("DB 1: \(chiffreMystère)")

for i in /* TODO */{
    print("Entrez un nombre:")
    let res /* Mettre le type de res */ = readLine()
    let nb = Int(res) // HMMMM Ce truc marche pas. Que faire?
    print("DB 2: \(nb)")

    if i > nbCoupsMax {
        print("Perdu")
        break // Faut-il mettre un break ici?
    } else if nb == chiffreMystère {
        print("Gagné")
        break
    } else {
        if nb /* TODO : Insérer un comparateur*/ chiffreMystère {
            print("Le nombre à deviner est plus grand.")
        } else {
            print("Le nombre à deviner est plus petit.")
        }
    }
}

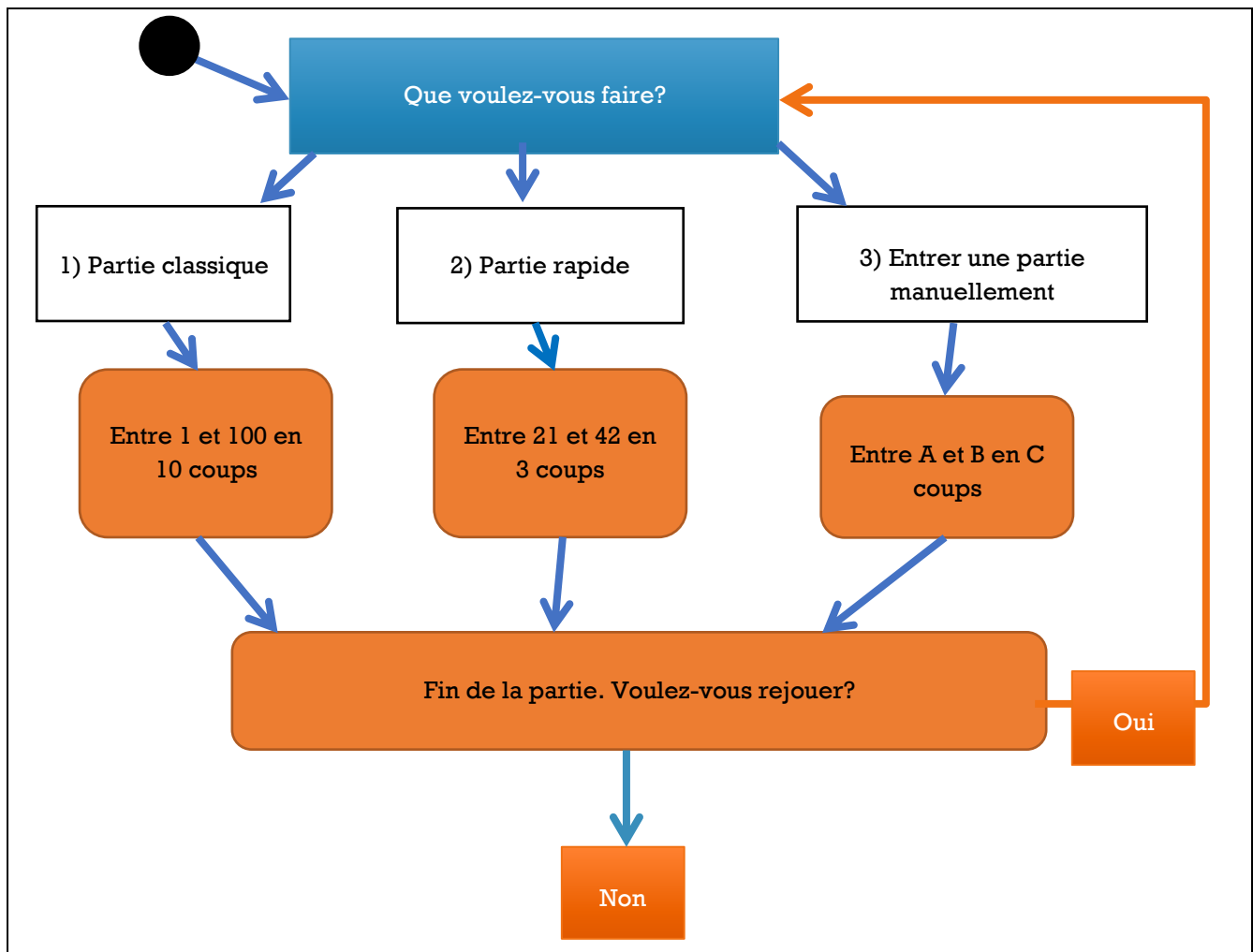
print("Bye")
```

ÉTAPE 1

1. Créer un projet pour terminal dans XCode (MacOs > Command Line Tool)
2. Copier le code ci-dessus dans le fichier main (Omettez les premières lignes de commentaire)
3. Débuguer le programme pour qu'il fonctionne pour des bornes entre 1 et 100 avec 10 coups

ÉTAPE 2

Faire un programme qui implante le diagramme d'utilisation suivant :



À la fin de chaque partie il faut afficher un message correspondant au résultat de la partie.

ÉTAPE 3

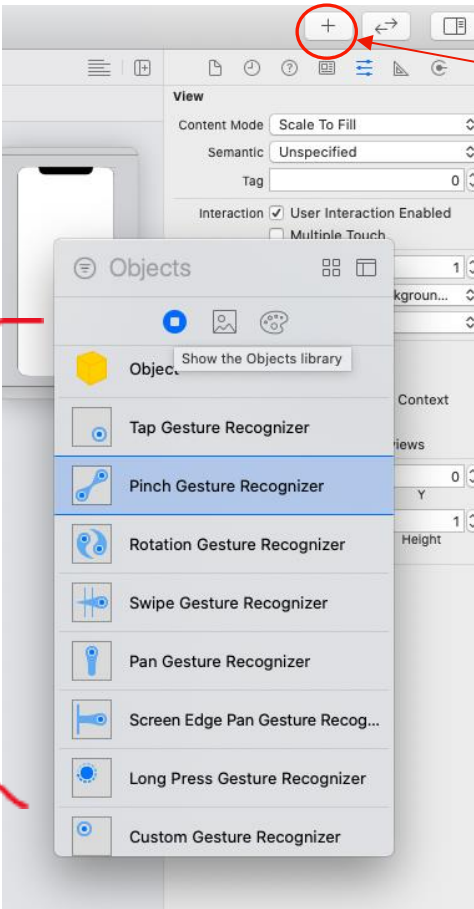
Si vous deviez faire une application avec une seule vue (« single view » ou monovue) de votre jeu après l'étape 2, quel serait votre design? Allez-vous modifier le diagramme d'utilisation afin de gérer la contrainte monovue?

Quels éléments XCode utiliseriez-vous afin d'offrir la meilleure expérience à vos utilisateurs?

Ajouter une capture d'écran, ou une photo, de votre design. Vous devez remettre une maquette (« mockup ») faites dans « interface builder », et vous pouvez en plus remettre un diagramme fil de fer, ou d'adobe XD.

BONUS :

Dans la librairie d'objets de Xcode combien y-a-t-il d'éléments? Précisez votre version de XCode et la version iOS du projet.



En tout et pour tout, combien y a-t-il d'éléments dans cette liste?

The screenshot shows the Xcode interface with the 'Objects' library open. A red circle highlights the '+' icon in the top right corner of the library, and a red arrow points to it. A red line connects this icon to an orange callout box on the left. The library lists the following elements:

- Object
- Tap Gesture Recognizer
- Pinch Gesture Recognizer
- Rotation Gesture Recognizer
- Swipe Gesture Recognizer
- Pan Gesture Recognizer
- Screen Edge Pan Gesture Recognizer
- Long Press Gesture Recognizer
- Custom Gesture Recognizer