OpenQuizz

Comme amélioration, j'ai opté pour une animation spécifique à la fin du quizz :

```
private func showQuestionView() {
    questionView.transform = .identity
    questionView.transform = CGAffineTransform(scaleX: 0.01, y: 0.01)

    questionView.style = .standard

    switch game.state {
    case .ongoing:
        questionView.title = game.currentQuestion.title
    case .over:
        questionView.title = "Game Over"
        animateScore()
        animateEnd()
    }
```

1. Ajout de méthodes dans la méthode ShowQuestionView :

J'ai ajouté à la méthode ShowQuestionView() deux méthodes quand le jeu est terminé pour cette animation spécifique, à savoir : animateScore() et animateEnd()

```
private func animateEnd() {
    if game.state == .over  {
        UIView.animate(withDuration: 2){
            self.questionView.alpha = 0
            self.questionView.transform = .identity
            self.questionView.alpha = 1
        }
    }
}
```

2. La méthode animateEnd() :

Elle permet à la fin du quizz de faire disparaitre la vue des questions puis elle réapparait avec Game Over. Pour cela j'ai utilisé **« Alpha »** dans le code.
Cette propriété permet de modifier l'opacité de la vue. En l'animant, on peut faire apparaître ou disparaître doucement notre vue.

```swift
private func animateScore() {
    if game.state == .over  {
        scoreLabel.transform = CGAffineTransform(scaleX: 2, y: 2)
        UIView.animate(withDuration: 1, delay: 0, usingSpringWithDamping:
            0.5, initialSpringVelocity: 0.5, options: [], animations: {
            self.scoreLabel.transform = .identity
        }, completion: nil)
    }
}
```

3. La méthode animateScore() :

Elle permet à la fin du quizz d'animer le score final avec une oscillation, pour ce faire j'ai utilisé les paramètres suivants :

- spring : Les animations spring permettent de créer un effet d'oscillation autour de la valeur d'arrivée de l'animation.

- damping : Ce paramètre peut être choisi entre 0 et 1. Plus on est proche de 0, plus il y aura d'oscillations autour de la valeur d'arrivée.

- initialVelocity : Cela permet de choisir la vitesse de départ de la vue lors de l'animation. Plus elle sera rapide, plus les oscillations seront grandes.