COSC2659: iOS Development

Assessment 2

Project Title: Deck of Destiny

Nguyen Nhat Lam - s3989101

Introduction

DECK OF DESTINY

Overview and Context

Deck of Destiny is a fantasy-themed 2D card game designed for iOS devices. The game combines elements of classic card games with modern UI/UX design, aiming to provide an engaging and visually appealing experience. The project aims to create an experience where players can immerse themselves in a unique card-based gameplay that includes various levels, leaderboards, and customization options.

The primary goal of the game is to challenge players with increasingly difficult levels while allowing them to track their progress and compete with others, many of the features implemented will reflect this. The target audience includes casual gamers who enjoy simple, yet strategic card games, as well as those looking for a fun and competitive experience on their iOS devices. As such cross-platform compatibility is also a goal to keep in mind for this target audience.

Acknowledgement: I acknowledge that the game assets are not my own and are royalty-free downloadable assets from creators cafeDraw and SnoopethDuckDuck on Itch.io.

How to Play the Game

Deck of Destiny is played by drawing cards from a deck and using them to achieve specific objectives or scores. The game rules include:

1. **Drawing Cards:** Players draw cards from the deck
2. **Objectives:** Match each cards together in order to win points. As well as avoid any traps and obstacles.
3. **Scoring:** Players earn points based on their performance and the cards they play.
4. **Levels:** The game features multiple levels with increasing difficulty and new challenges.
5. **Endgame:** The game ends when a player runs out of cards.

Motivation and Inspiration

The concept for **Deck of Destiny** was inspired by classic fantasy card games such as Hearthstone and the desire to create a digital version that is both engaging and easy to pick up. The idea was to combine traditional gameplay mechanics with modern features like leaderboards and customizable settings to appeal to a wide range of players.

Technical Features

1. Menu View

```
import UIKit

class MenuViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // Adding a logo image at the top
        let logoImageView = UIImageView(image: UIImage(named: "GameLogo")) // Replace with your image name
        logoImageView.contentMode = .scaleAspectFit
        logoImageView.frame = CGRect(x: 50, y: 100, width: 300, height: 200) // Adjust frame as needed
        view.addSubview(logoImageView)

        setupUI()
    }
```

```swift
func setupUI() {
    // Set background color
    view.backgroundColor = .white

    // Create a label for the game title
    let titleLabel = UILabel()
    titleLabel.text = "Deck of Destiny"
    titleLabel.font = UIFont.systemFont(ofSize: 32, weight: .bold)
    titleLabel.textAlignment = .center
    titleLabel.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(titleLabel)

    if GameData.shared.loadGameState() != nil {
        let continueButton = createButton(title: "Continue", action: #selector(continueGame))
        // Add the button to the view hierarchy
        // Adjust constraints and layout
    }

    // Create buttons for navigation
    let gameButton = createButton(title: "Play Game", action: #selector(startGame))
    let leaderboardButton = createButton(title: "Leaderboard", action: #selector(showLeaderboard))
    let howToPlayButton = createButton(title: "How To Play", action: #selector(showHowToPlay))
    let settingsButton = createButton(title: "Settings", action: #selector(showSettings))

    // Add buttons to the view
    let stackView = UIStackView(arrangedSubviews: [gameButton, leaderboardButton, howToPlayButton, settingsButton])
    stackView.axis = .vertical
    stackView.spacing = 20
    stackView.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(stackView)
```

Implementation: The Menu View as the screenshots of the code displays, includes a logo image and as well as features to add buttons for navigating to different parts of the app, such as the Game View, Leaderboard View, How To Play View, and Settings View. Buttons are added programmatically with custom styles and actions to handle navigation.

2. Settings View

```
func setupUI() {
    view.backgroundColor = .white

    // Difficulty Picker
    difficultyPicker = UIPickerView()
    difficultyPicker.delegate = self
    difficultyPicker.dataSource = self
    difficultyPicker.tag = 1
    difficultyPicker.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(difficultyPicker)

    // Language Picker
    languagePicker = UIPickerView()
    languagePicker.delegate = self
    languagePicker.dataSource = self
    languagePicker.tag = 2
    languagePicker.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(languagePicker)

    // Setup Constraints
    NSLayoutConstraint.activate([
        difficultyPicker.centerXAnchor.constraint(equalTo: view.centerXAnchor),
        difficultyPicker.centerYAnchor.constraint(equalTo: view.centerYAnchor, constant: -100),

        languagePicker.centerXAnchor.constraint(equalTo: view.centerXAnchor),
        languagePicker.centerYAnchor.constraint(equalTo: view.centerYAnchor, constant: 100)
    ])
}
```

Implementation: The Settings View allows for toggle between a variety of difficulty settings, from Easy to Medium and finally to Hard. Users may also toggle between light and dark themes and customize game settings. Images are used for theme icons, and switches are implemented to handle user preferences. There are also language features for localization purposes.

3. Leaderboard View

```swift
import UIKit

class LeaderboardViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {

    var tableView: UITableView!
    var scores: [(username: String, score: Int)] = [("Player1", 100), ("Player2", 80), ("Player3", 60)]

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

    func setupUI() {
        view.backgroundColor = .white

        // Table View
        tableView = UITableView()
        tableView.dataSource = self
        tableView.delegate = self
        tableView.register(UITableViewCell.self, forCellReuseIdentifier: "cell")
        tableView.translatesAutoresizingMaskIntoConstraints = false
        view.addSubview(tableView)

        // Setup Constraints
        NSLayoutConstraint.activate([
            tableView.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor),
            tableView.leadingAnchor.constraint(equalTo: view.leadingAnchor),
            tableView.trailingAnchor.constraint(equalTo: view.trailingAnchor),
            tableView.bottomAnchor.constraint(equalTo: view.bottomAnchor)
        ])
```

Implementation: The Leaderboard View displays players' scores and their rankings in a simple table as well as achievement badges. Images represent badges, and labels show usernames and scores. This view also includes graphical representations of player statistics using bar and line charts.

4. How To Play View

```
// Instructions Label
let instructionsLabel = UILabel()
instructionsLabel.text = """
1. Draw cards from the deck.
2. Match cards to win points.
3. Avoid traps and obstacles.
4. The game ends when you run out of cards.
"""
instructionsLabel.font = UIFont.systemFont(ofSize: 18)
instructionsLabel.numberOfLines = 0
instructionsLabel.textAlignment = .left
instructionsLabel.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(instructionsLabel)
```

Implementation: This view provides an interactive tutorial with images explaining each step of the game. Images are used to illustrate gameplay mechanics and tips (to be implemented and developed later).

5. Game View

```swift
import UIKit

class GameViewController: UIViewController {

    var currentScoreLabel: UILabel!
    var gameBoardView: UIView!
    var gameOverLabel: UILabel!
    var gameState: GameState!

    var score = 0 {
        didSet {
            currentScoreLabel.text = "Score: \(score)"
        }
    }

    let themeSwitch = UISwitch()

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

        if let savedState = GameData.shared.loadGameState() {
            gameState = savedState
            resumeGame()
        } else {
            startNewGame()
        }
    }

    func setupUI() {
        view.backgroundColor = .white

        // Current Score Label
        currentScoreLabel = UILabel()
        currentScoreLabel.text = "Score: 0"
        currentScoreLabel.font = UIFont.systemFont(ofSize: 24)
        currentScoreLabel.translatesAutoresizingMaskIntoConstraints = false
        view.addSubview(currentScoreLabel)
```

```swift
func startGame() {
    // Initialize game logic here
    score = 0
    gameOverLabel.alpha = 0
}

func startNewGame() {
    gameState = GameState(score: 0, currentLevel: 1)
    loadLevel(level: gameState.currentLevel)
}

func loadLevel(level: Int) {
    // Adjust difficulty settings based on the level
    switch level {
    case 1:
        // Easy level settings
        break
    case 2:
        // Medium level settings
        break
    case 3:
        // Hard level settings
        break
    default:
        // Higher levels
        break
    }
}

func resumeGame() {
    // Resume game using `gameState`
}

func endGame() {
    // Display game over message
    UIView.animate(withDuration: 1.0) {
        self.gameOverLabel.alpha = 1.0
    }
}
```

Implementation: The Game View is the main gameplay screen where players interact with the game. It includes a card image and score display. Animations are used to enhance the gameplay experience.

Known Bugs/Problems

- Many features are still missing and underdeveloped.
- Animation codes do not run properly and are underdeveloped.
- Leaderboards do not always display properly.
- Occasionally, images may not load correctly on specific devices.
- There may be minor layout issues on different screen sizes.

Design Elements and User Experience

Visual Appeal

Visual Appeal

The design of Deck of Destiny is crafted to be both visually appealing and user-friendly. In light mode, the app features a clean, bright interface with a white background and dark text. This choice enhances readability and ensures that the content stands out clearly. The color scheme is designed to be both eye-catching and accessible, with high contrast between text and background to support users in various lighting conditions. Conversely, the dark mode presents a more modern and sophisticated appearance with a dark background and light text. This mode is not only aesthetically pleasing but also helps reduce eye strain during prolonged gameplay sessions, catering to user comfort.

The use of color schemes is carefully considered to create a cohesive and engaging visual experience. For instance, the color palette for buttons, icons, and other UI elements is selected to provide clear visual cues and maintain consistency throughout the app. The design ensures that users can easily distinguish between interactive elements and non-interactive content, enhancing the overall usability of the app.

Intuitive User Interface

The user interface (UI) of Deck of Destiny is designed with a focus on simplicity and intuitive navigation. Buttons, menus, and other interactive elements are strategically placed to ensure ease of use. The main menu provides clear options for starting the game, accessing the leaderboard,

viewing game instructions, and adjusting settings. Each view is designed to be self-explanatory, minimizing the need for extensive instructions and allowing users to navigate the app seamlessly.

Immediate feedback is an essential aspect of the UI design. When users interact with buttons or other elements, visual and textual feedback is provided to confirm their actions. This responsiveness not only helps users understand the results of their actions but also keeps them engaged with the app.

Usability considerations are central to the design, with a layout that prioritizes user needs. For example, the settings view allows users to easily adjust preferences and customize their experience, while the leaderboard view provides a clear overview of high scores and achievements. The app's design is focused on making gameplay and navigation straightforward, ensuring that users of all ages and skill levels can enjoy the game without frustration.

User-Centered Design

The design of **Deck of Destiny** places a strong emphasis on user-centered principles. The app is designed to prioritize the needs and preferences of its users by offering customization options and ensuring accessibility. The settings view, for example, allows users to switch between light and dark themes and adjust difficulty settings to tailor the gameplay experience to their preferences.

Accessibility is a crucial consideration in the app's design, which is why there is a strong emphasis on localization. The interface is also designed to be user-friendly for a diverse audience, including those with varying levels of experience and different physical abilities. Features like clear text labels, high-contrast colors, and intuitive navigation contribute to making the app accessible to all users.

Conclusion

Reflection

Developing Deck of Destiny provided valuable experience in working with UIKit and Swift, something I have little experience before learning the language and framework. The experience was primarily focusing on creating a seamless user experience and implementing core game features that required some critical thinking and problem solving. The project showcased the integration of various elements, from UI design to gameplay mechanics that was difficult to implement at times. As such I feel I still have many shortcominsg during the project and that I still have a lot to learn in the development of this application.

Future Improvements

Potential improvements include:

- **Enhanced Graphics:** Adding more detailed graphics and animations that function properly. Allow time to develop original assets.
- **Developed Features:** Fixing all the issues that came with the lack of features.
- **Expanded Gameplay:** Introducing new game modes and challenges.
- **Bug Fixes:** Resolving known bugs and optimizing performance.

Reference

- **Assets:** Royalty-free downloadables by creators cafeDraw and SnoopethDuckDuck on Itch.io.

- **Swift Documentation:** Apple Developer
- **UIKit Documentation:** Apple Developer UIKit