AR Measure iOS App

Final Project: CS-8395 Augmented Reality

Github Link: https://github.com/guptavaibhav35/AR-Measure-App

Project Goal

The primary goal of the AR Measure App project is to create an accessible, user-friendly mobile application that enables users to measure physical objects and distances in their immediate environment using the power of augmented reality (AR) technology. The application intends to simplify tasks that traditionally require a physical measuring tape or ruler, by digitalizing the measurement process through a smartphone's camera interface.

The project aims to achieve the following detailed objectives:

1. **Augmented Reality Integration**: Utilize AR technology to overlay digital information onto the real world as seen through the phone's camera. This will be the core feature allowing users to perceive and interact with digital measurements as if they were part of the physical space around them.

2. **Ease of Use**: Provide an intuitive user interface that can be navigated easily by users of all ages and backgrounds. The app should require minimal learning curve, encouraging widespread adoption and convenience.

3. **Accuracy**: Ensure that the measurements provided by the app are accurate and reliable. This will involve precise calibration and robust development to maintain the integrity and utility of the app's measurements.

4. **Cross-Platform Availability**: Although initially targeting iOS, the structure of the app should be such that it allows for future expansion to other platforms, ensuring the broadest possible user base.

5. **Real-Time Measurements**: Render real-time measurements on the device's display as the user points their camera at objects, offering instant feedback.

6. **User Interaction**: Implement user-friendly ways to start and end measurements, such as tapping on the touch screen to place measurement markers in the AR space.

The accomplishment of these objectives is expected to result in a practical and innovative tool that harnesses the advantages of AR to simplify everyday measurement tasks. By combining technology with utility, the AR Measure App project aspires to modernize the way measurements are taken and to become an indispensable tool in the toolkit of professionals and hobbyists alike.

Design Approach

The design process for the AR Measure App was guided by a user-centric philosophy. The following steps were taken to ensure the application was tailored to the needs and preferences of the end-user:

Technical Design

The technical design was approached with a focus on reliability, performance, and scalability:

1. **Selection of AR Framework**: Unity's AR Foundation was chosen as the AR framework due to its compatibility with ARKit and the potential for cross-platform development.

2. **Modular Architecture**: The app's architecture was designed with modularity in mind to facilitate maintenance and future feature additions. Individual components were encapsulated to minimize dependencies and allow for easier updates and testing.

3. **Performance Optimization**: Efficient algorithms and optimizations were employed to ensure smooth performance all iOS devices, keeping in mind the varying processing powers and camera specifications.

4. **Calibration and Accuracy**: A calibration procedure was designed to ensure measurement accuracy, including the establishment of a baseline measurement to calculate real-world distances.

5. **Use of Visual Cues**: The implementation of visual cues (e.g., lines, anchors) in the AR environment was carefully designed to assist users in understanding the measurement process and to enhance the accuracy of placed measurement points.

The design of the AR Measure App was a cohesive effort between understanding the users' needs, leveraging the latest in AR technology, and ensuring a robust and flexible technical foundation. The ultimate goal was to deliver an application that not only met but exceeded expectations in terms of usability and performance.

Technical Design, Architecture and Development Process

Unity Engine and AR Foundation

The AR Measure App's foundation is the Unity engine, known for its strong capabilities in AR development. Unity's extensive set of features allows for the creation of rich, immersive AR experiences. AR Foundation, a package provided by Unity, facilitates cross-platform AR development and was chosen for its ability to work with ARKit, enabling our iOS-focused release.

AR Foundation provides a generalized set of functionalities such as plane detection, anchor creation, and AR session management critical to our measurement app's functionality.

Shader Development

Custom shaders were created using Unity's ShaderLab and HLSL to define the aesthetic representation of the measurements within the AR environment. These shaders are responsible for the rendering of measurement lines, interactive points, and text overlays that integrate with the live footage from the iOS device's camera, providing users with a clear, visually-appealing interface.

Technical Components

1. **Scene Management**: The app's lifecycle is managed by different scenes within the Unity project, providing a seamless user experience from start to finish.

2. **AR Camera**: The AR camera emulates the movement and viewpoint of the real-world, anchoring digital content to physical locations as the user moves their device.

3. **UI and Interaction**: Unity's integrated UI toolkit is utilized to deliver a responsive user interface that adjusts to different screen sizes and resolutions of iOS devices. Touch and gesture detection are employed to interact with AR elements within the environment.

4. **Measurement Logic**: Developed in C#, the core measure logic converts distances between AR anchors from Unity units to real-world metrics, accounting for device calibration to maintain accuracy.

5. **iOS Build Configuration**: Unity's build settings are tailored for the iOS platform, with adjustments made to aspects such as the bundle identifier, necessary permissions (like camera access), and other settings to fine-tune the app for iOS devices.

6. **Performance Optimization**: Various optimization strategies, such as occlusion culling, LOD adjustments, and memory management are implemented to ensure a smooth AR experience, a crucial factor when engaging in real-time computation and rendering.

Thoroughly engineered, each technical aspect is carefully integrated within the Unity framework to forge the AR Measure App as a benchmark for precision and user-friendliness in mobile AR measurement tools for iOS platforms.

Novelty of the AR Measure App

The AR Measure App stands out with its innovative features and enhancements, setting it apart from conventional measurement tools and other AR measurement apps. Here are some of the key innovative features tailored for iOS:

1. **Intuitive Real-World Interaction**: Leveraging ARKit's capabilities, the app allows users to intuitively interact with their environments. Placing virtual markers with simple screen taps integrates digital measurements naturally into the physical world for a seamless user experience.

2. **Advanced Calibration for Accuracy:** The app goes beyond standard accuracy with a sophisticated calibration process adapted for iOS devices. It considers the unique camera

specifications and environmental variations, delivering precise measurements users can depend on for exacting tasks.

3. **ARKit-Focused AR Foundation Use**: Being developed on Unity's AR Foundation with a focus on ARKit, the app ensures a refined AR experience on iOS devices. This approach lays the foundation for a robust and versatile AR application that maximizes the advanced features ARKit offers.

These pioneering features demonstrate the AR Measure App's commitment to harnessing the power of iOS's AR technology for practical applications. A focus on precision, user engagement, and intuitive design places this app at the vanguard of iOS AR measurement solutions.

Limitations

Despite the careful design and advanced features included in the AR Measure App for iOS, there are inherent limitations within the application that users and developers should be aware of:

Hardware Dependency

1. **Device Compatibility**: The app's performance and capabilities are heavily reliant on the user's device supporting ARKit. Older iOS devices that do not support ARKit will not be able to run the application.

2. **Sensor Accuracy**: The precision of the measurements can vary based on the quality of the device's sensors. While ARKit provides a robust platform for AR applications, inconsistencies in sensor data can lead to slight inaccuracies in measurement.

Environmental Conditions

1. **Lighting Requirements**: For ARKit to function optimally, the app requires good lighting conditions. Poorly lit environments may result in decreased tracking stability and measurement accuracy.

2. **Surface Detection**: The application is designed to detect horizontal and vertical planes, but complex or highly reflective surfaces can pose challenges for the AR technology, potentially affecting the app's functionality.

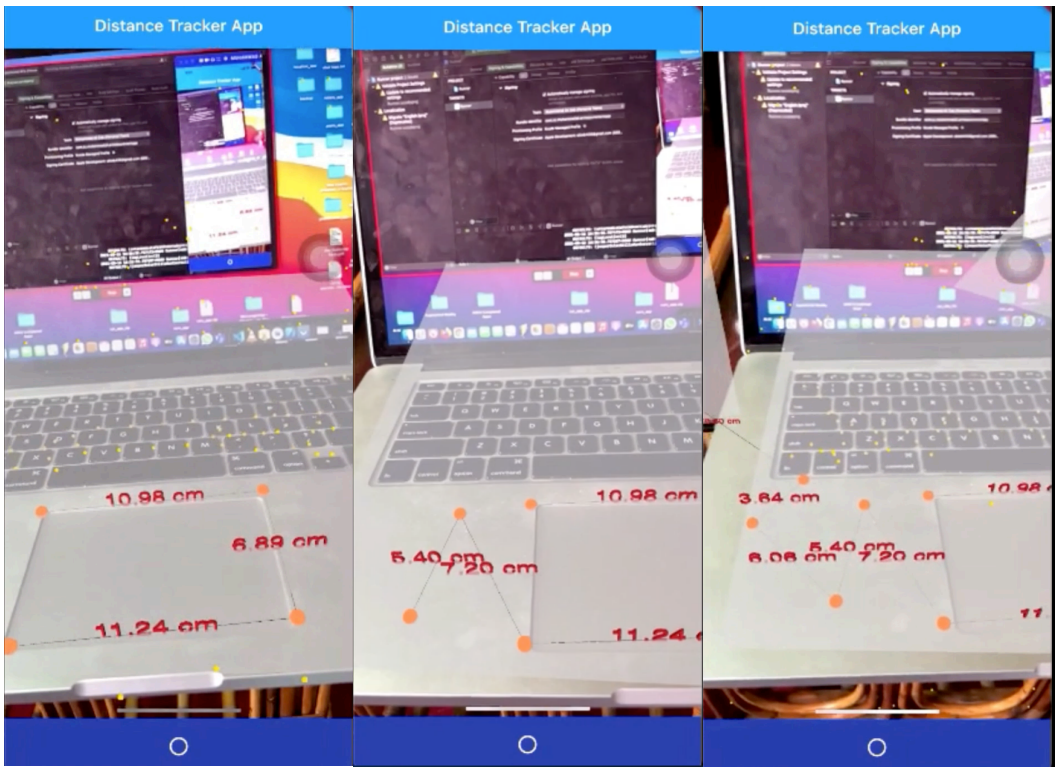
Software Constraints

1. **Operating System Dependency**: The app requires a certain minimum version of iOS to run. Users with devices not updated to this version or higher would be unable to install or use the app.

2. **ARKit-Specific Features**: Development choices made to take advantage of ARKit features may limit the introduction of certain functionalities that are not supported natively by ARKit.

Identifying these limitations is an important step towards ongoing improvement and helps set realistic expectations for end-users. It also provides a roadmap for future development, ensuring that the app remains a leading choice for AR measurement on iOS devices.

Images



References

1. **Apple Developer Documentation:** https://developer.apple.com/documentation/arkit
2. **Measuring Real-World Distances in an AR Experience:** https://developer.apple.com/documentation/arkit/measuring_real-world_distances_in_an_ar_experience
3. **Unity AR Foundation Documentation:** https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@latest
4. **SceneKit Documentation**: https://developer.apple.com/documentation/scenekit