“Style My Space 3D”

SUBMITTED TO THE MSBTE, MUMBAI IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE AWARD

DIPLOMA IN INFORMATION TECHNOLOGY

BY

1. **Padhiyar Tejas R. (1800340523)**
2. **Bhadke Mansi M. (1800340348)**
3. **Shete Nikita N. (1800340564)**
4. **Patni Harsh P. (1900340537)**

Under the Guidance of

Miss. R. S. Patil

DEPARTMENT OF INFORMATION TECHNOLOGY

SANJIVANI RURAL EDUCATION SOCIETY’

S.K.B.P. POLYTECHNIC, KOPARGAON

DIST. AHMEDNAGAR - 423603

(2020-21)

Department of Information Technology

S.K.B.P. POLYTECHNIC, KOPARGAON

CERTIFICATE

THIS IS TO CERTIFY THAT PROJECT ENTITLED

“Style My Space 3D”

Submitted by

Padhiyar Tejas R.

Bhadke Mansi M.

Shete Nikita N.

Patni Harsh P.

Is a bonafide work carried out by above students under the supervision of Miss. R. S. Patil and it is submitted towards the partial fulfilment of the requirement of MSBTE, Mumbai for the award of Diploma in Information Technology.

(Miss. R. S. Patil)
Project Guide

(Miss. R. S. Patil)
Project Coordinator

(Mr. K. P. Jadhav)
HOD, IT Department

(Mr. A. R. Mirikar)
Principal

Date: / /

Place: Kopargaon.

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI

EXAMINERS CERTIFICATE

"Style My Space 3D"

Submitted by

1. **Padhiyar Tejas R.**
2. **Bhadke Mansi M.**
3. **Shete Nikita N.**
4. **Patni Harsh P.**

For the partial fulfilment of the requirement of Diploma in Information Technology examined & certified.

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

This minor project would not have been possible without the valuable assistance of many people to whom we are indebted, in particular, our project coordinator Miss. R. S. Patil of Sanjivani K.B.P. Polytechnic College.

We would also like to thank "Department of IT", Sanjivani K.B.P. Polytechnic College for providing us with the necessary components for our project. We also thank to all the teachers of Information Technology Department who helped us in many difficult situations regarding the project and provided with the necessary advice.

We also would like to thank HOD of "Department of IT", Sanjivani K.B.P. Polytechnic College for providing us space to do our project and special thanks to the members Padhiyar Tejas R, Bhadke Mansi M, Shete Nikita N, Patni Harsh P. A special word of thanks is to our class mates and our families for providing us the moral support.

Submitted By: -

Padhiyar Tejas R.

Bhadke Mansi M.

Shete Nikita N.

Patni Harsh P.

INDEX

Abstract

Building a dream house is a lifelong desire for many. After years of savings, planning and hard work, an individual or a family decides to build their dream house. Interior design is a process that provides its customers with a set of aesthetically pleasing but efficient solutions for a better use of the space in question. The goal of interior design is to improve the user experience by better managing the space available in the intervened environment.

In today's growing and emerging world of new technologies where new innovations are regularly done to make our lives easier, Augmented Reality is one of the fastest growing fields, with applications in almost all industries, it will be considered as one of the key technologies of the future. It is safe to assume that interior design will be one of the biggest industries to get revolutionized by AR. Since the launch of Pokémon Go in 2016, augmented reality has been the talk of the mobile IT industry. Three years later, it's not limited to gaming but is part of a wide range of apps in a selection of business niches. Home decor businesses are among those that can profit from an AR-supporting mobile app. AR is a computer-generated content, overlaid on the real-world environment. Industries including healthcare, education, real-estate, farming, and broadcasting are already using this technology.

It often becomes challenging to visualize a piece of furniture, colour of the walls and the floor map for a new project. The mood boards are no more in fashion now. A new age customer wants a customized or personalized solution exactly according to their needs. Augmented reality in interior design is the revolutionary solution to the constraints faced by clients and designers in visualizing the actual project like never before.

Nowadays with excessive work load and busy life, many professional face problems that results in the loss of their clients or the certain overheads that spoil the process of satisfying the client. The proposed software will be used by interior designers or architects. This proposed research most likely acts as an effective tool which can decrease the gap between industrial company and customer in addition to other applicable business communities. It will help in visualizing architect plans and interior designs. A virtual model of real environment can be designed before its physical implementation, it will allow interior designers to implement their idea in the given workspace virtually and then view it in real environment, it will also allow architects to view their 3D visualizations on their 2D drawings.

1. Introduction

Did you know that modern technology can influence the home design area immensely? Augmented Reality (AR) is one of the biggest technology trends, combining a real-world environment with computer-generated input. The Interior designing app is based on AR technology.

Since the advancements in Computer Vision technologies and the improvements in smartphone cameras, AR has evolved into a massive $18.8 Billion industry. A lot of applications of AR are possible right now but the most hyped and the most successful industry so far is the use of AR in interior design and decoration. Recently augmented reality (AR) furniture arrangement systems help users overlay virtual furniture onto the real world. Such systems allow people to see how the room will look with new furniture without actually buying or moving real furniture. Augmented Reality (AR) is a kind of Computer Vision Technology which can add the virtual information to the real environment, which makes the virtual and the real as a whole. It is a popular research area in recent years, more and more researchers studied on it.

The eyes of humans have the capability of seeing objects in the third dimension (3D) but, imagining how your future home is going to look by just seeing a picture is difficult for many people. The use of Augmented Reality thus came into the scene. Augmented Reality revolves around overlaying virtual objects in the real environment. This also provides its user the ability to interact with the virtual object and set it according to their like. Augmented Reality is one booming technology which is helping the Architect to perfectly connect with the client and users. This immersive technology can be used effectively to creatively showcase the model to the clients. It can help them instantly by modifying changes according to the client's need. AR in architects and designers are gradually integrating AR experience in their business. This helps in the promotion of their ideas, for the customers to feel confident before they buy. Thus, the potential of this technology beneficial for this sector on a vast scale.

1.1 Problem Statement:

Interior design involves a lot of people management; in addition to finding the right partners inside and outside firm, they'll also have to work with their clients often quite closely. Many clients come to interior designers because they have visions, but not the resources to execute them. Others may not have a particular design in mind, but they might be alarmed by costs. Still other clients will ask you to achieve two contradictory ends or have unreasonable expectations about timelines. Managing your clients' expectations is a huge part of interior design—and it's always challenging, no matter how large or small your business is. So, it's really important for designer to visualize idea of clients and fulfil their requirements.

Augmented Reality has the potential to have big impact on the interior design industry. This industry thrives on the choices given by designers and the choices made by the user. There are a lot of variables like choosing a colour that matches with the room, size of the room, how a certain piece of furniture would blend with the room. Hence making a decision gets very difficult. The biggest problem is visualizing how a piece of furniture is going to look at a certain place. Even though the designers give a visual representation by showing rendered images of that room but it isn't immersive. This is where AR gets a higher ground. It provides high level of immersion to the user by using 3D models and depth sensing compared to traditional 2D image. The user also has the ability to interact and move around the object to place it according to their like Furniture retail is a huge market in the world with a Val AR helps for better visualization, thus customer's views & ideas can be visualized providing more clear design. So, it can be changed quickly if customer wants some changes. This will not leave customer unsatisfied at the end. Time management is really important for interior designers by such approaches time can be saved at great extent.

1.2 Aim of Proposed System

AR in interior design help the client visualize the project before it is developed. Using AR in interior design gives an ability to the user to design the space the way they want. Be it a rustic, beach or boho style home, AR will let the user visualize that all in a go. A mobile phone or a tablet loaded with samples is all needed to transform the user space. Using AR, it would be much easier for designers to present their innovative ideas to clients. A design idea board may not be enough for the clients with dynamic needs.

Interior Designers will get chance to make their ideas into real like objects using AR which will be better understandable by the customers. Ideas and views of designer will be more cleared. This will give less changes of fault in real work.

2. Need And Present Scope

- **Need: -**

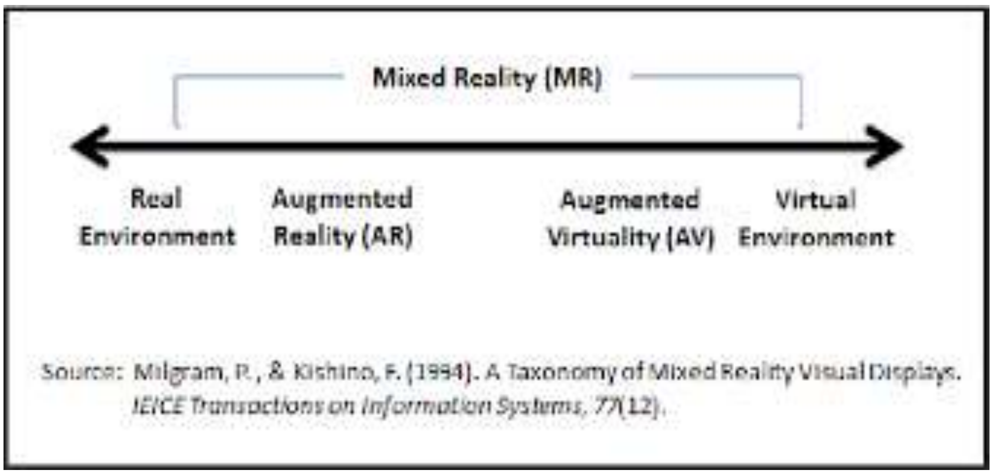
  Every individual has a dream to own a house for themselves and want to decorate and make their home colorful and this is possible only with the help of interior designers, that they understand the need of the customer and bring their dream home alive. Only interior designer can make a home-interior remarkable. aim of this project is to make the designers idea into real like object using AR which will be better understandable by customer, by making ideas and views of designer clearer. It is really very good idea to obtain a better visualization of the project for clients. This system can help the user to virtually try on object with their smartphone with personalized instructions and it is easier to fit into our daily lives due to the utilization of smartphones. 3D or virtual model, not only speed up the design process but also enables architects or designer to play around with different ideas and identify potential design problem before they become actual issue. Using this system, in designing, by virtually putting all the pieces of furniture together, provide a real view of complete project. It provides a powerful additional capability to the business that complete the user’s experience. It is smart way to connect potential users in unique way.

- **Present Scope: -**
  1. Realistic, easy and quick.
  2. An Image speaks a thousand words.
  3. Less re-work.
  4. Increased productivity.
  5. Better for marketing and Project approval.
  6. Easier to fit into our daily lives.

3. Literature Survey

3.1. Augmented Reality

Augmented reality is the augmentation of human sensory perception by superimposing virtual objects on the real world, which is a variation of virtual environments (Azuma, 1997). As illustrated below in the Virtuality continuum (Fig. 1), AR sits between reality and complete virtuality that is called Virtual Reality.



Mixed Reality (MR)
Real Environment
Augmented Reality (AR)
Augmented Virtuality (AV)
Virtual Environment
Source: Milgram, P., & Kishino, F. (1994). A Taxonomy of Mixed Reality Visual Displays. IEICE Transactions on Information Systems, 77(12).

Figure 1:Reality Virtuality Continuum (Milgram, Milgram, Colquhoun & Colquhoun, 1999)

There are several ways to display AR according to Brosch art and Zeile (Brosch art & Zeile, 2014):

- Projective Augmented Reality (PAR) where digital information is projected onto the real world and the user does not need to use any visual aids
- Video See-Through (VST) where projection glasses are needed to display content for the user's eyes
- Optical See-Through (OST) where a semi-transparent mirror is generally used augment the environment. Users use their own eyes to perceive the content
- Monitor Augmented Reality (MAR) which s a version of AR that needs a screen to display material, a rendering unit, and a camera

The display method used in this research can be referred to as an Advanced Monitor Augmented Reality, which uses tablets and smartphones similar to the study of Wang (Wang et al., 2014). It allows users to move freely in the real environment.

3.2. AR with Interior Design

Just as in many other fields, interior design is another one where researchers implement AR technologies. For instance, with Ikea's augmented furniture catalog, people became familiar with AR; however, in the background, AR needs a lot of trials to be effectively used in interior design. Siltanen & Karvonen presented a diminished reality solution aimed at interior design applications (2014). Another application by Wang allowed students to investigate specific buildings and their various systems with additional information using a phone or tablet (Wang et al., 2014). The IOS application Magic plan has gained a lot of attention as a leading tool for architectural and interior design for use in creating floor plans quickly, dimensioning interiors and developing 2D Plans using mobile sensors (Vaai & Vaai, 2014). Also, Unity and SketchUp programs enable the user to venture into the building as a walkthrough. The first-person controller is an asset that allows the user to explore a building. AR has been seen as a way to present projects and show a better understanding of customers' needs. "The augmented reality technology is the way of interior design future so that the cooperation of designers and consumer can be convenient and efficient" (Hui, 2015).

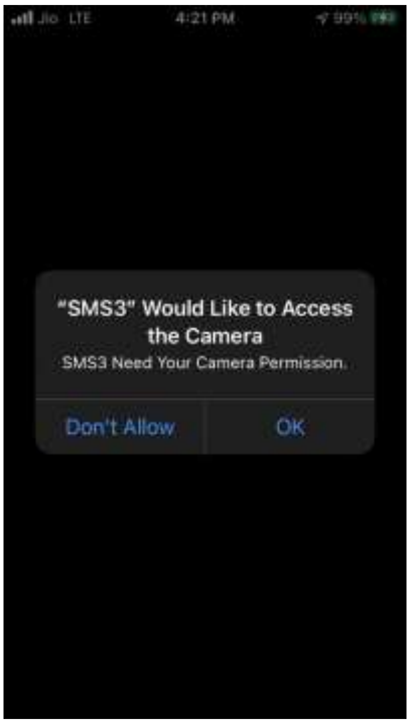
4. Working

The System Control View is consisting of the ControlButtonBar at the bottom with Browse Button at the middle, Setting Button at the right corner and the Most Recently placed button at left corner and the ControlVisibilityToggle button at the top right corner. Spacer at the center.

Working of the proposed system is as per given:

Step 1-Camera Permission:

Once the user downloads the app it will ask for the camera permission, to scan the place where he/she would like to visualize an object. The scanned surface will appear at the Spacer at the center of the screen.

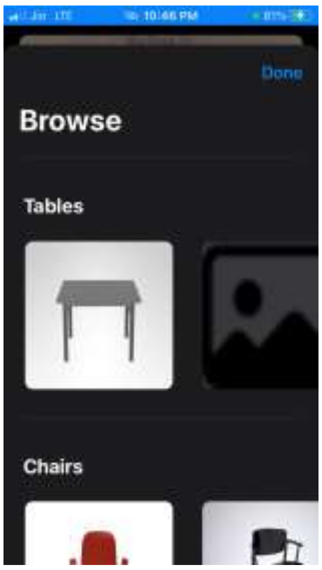

"SMS3" Would Like to Access the Camera
SMS3 Need Your Camera Permission.

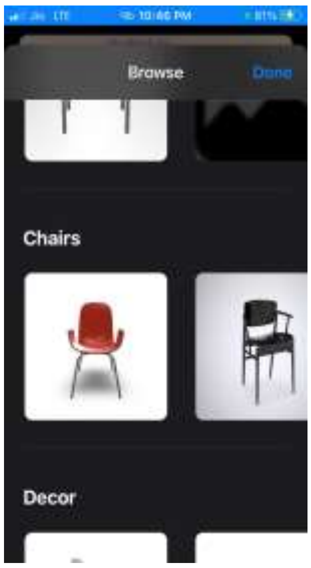
Step 2-Clicking on Browse Button:

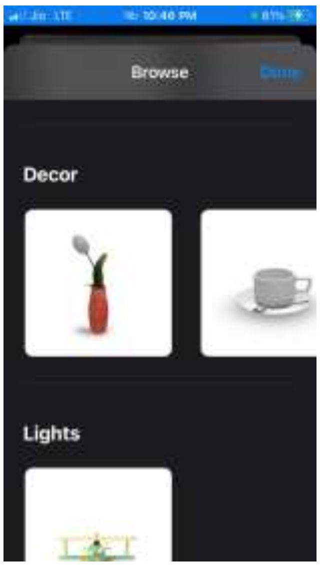
User must select object by clicking on browse button which is at the middle of the ControlButtonBar, once he/she click on the browse button, objects will appear on the Browse Panel which are divided into several categories such as Chair, Table, etc.

Step 3-Selecting the Object:

The user can select the object from the BrowsePanel as per his/her need or choice, and insert it at scanned surface and move it accordingly.

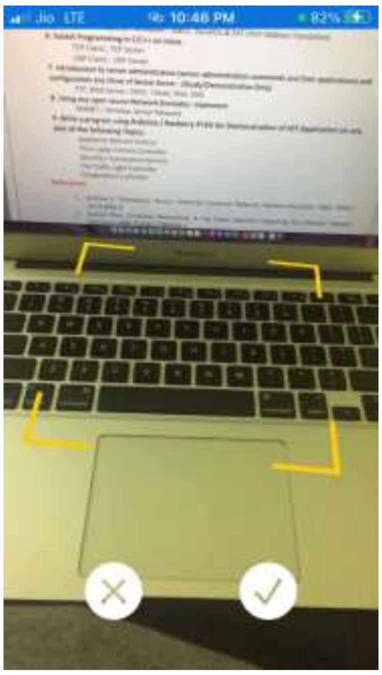
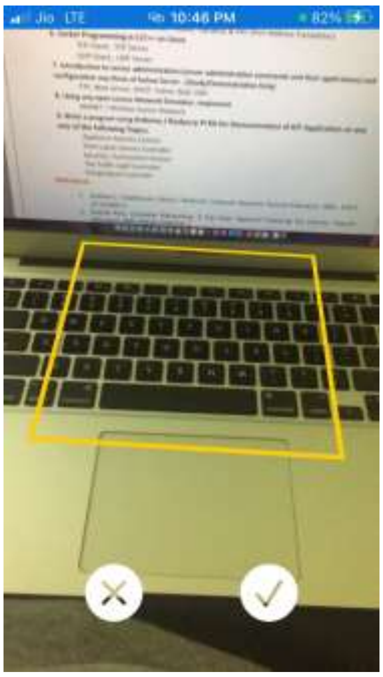
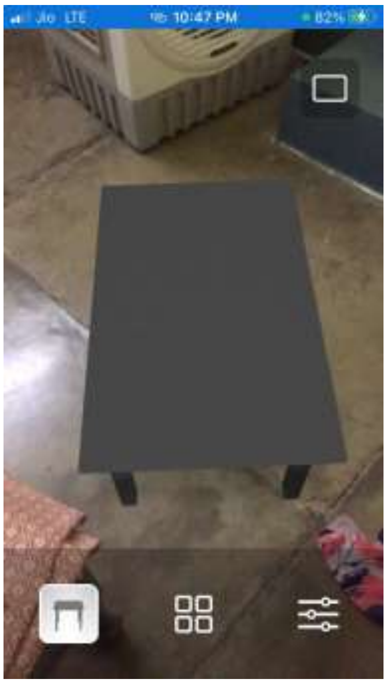






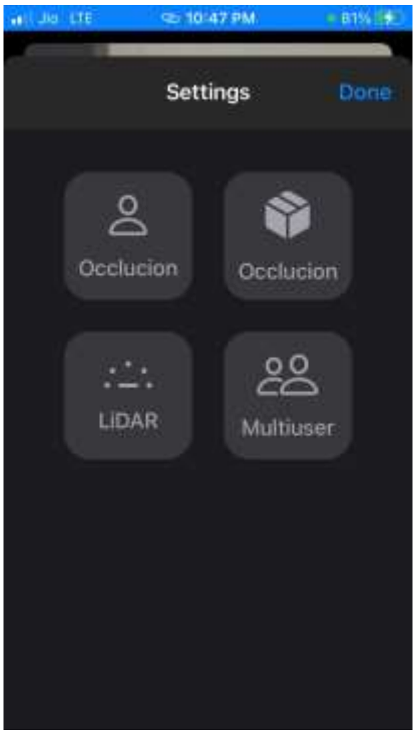
Step 4-Pacing the Object:

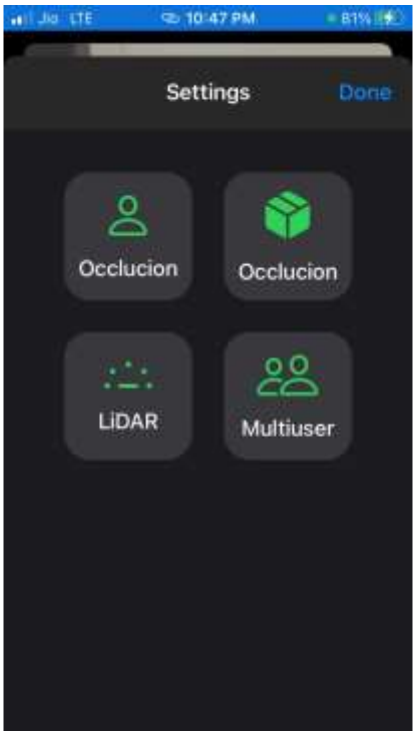
Then the user needs to focus on the surface where he/she want to place the selected object with the help of Focus Entity. After selecting object user can see Focus Entity square and by clicking on confirm button object will placed at the location of the Focus Entity. After placing object focus entity will get hidden.

Step 5- Use of additional Features:

Once the objects get placed, user can click on Settings Button at bottom right corner to enable/disable advanced features such as People Occlusion, Object Occlusion, LiDAR and Multiuser.





a. People Occlusion:

People Occlusion means that virtual object can be hidden or 'occluded' by any real-world objects. It means that if a real-world object (like a person) stands between users' device's camera and a virtual object, the virtual object will become hidden behind the real-world object.

Once the user enables People Occlusion option, the app can then render a virtually placed object behind the people who pass in front of camera.

b. Object Occlusion:

Object Occlusion often occurs when two or more objects come too close and seemingly merge or combine with each other.

Once the user enables Object Occlusion option, the app can then render a virtually placed object behind the physical object which is placed in front of camera.

c. LiDAR:

LiDAR stands for Light Detection and Ranging, is a remote sensing method that uses light in form of a pulsed laser to measure ranges (variable distance) to the Earth.

When the user enables the LiDAR option, LiDAR sensor scan the surface from side to side with pulsed laser beam as objects moves. And it will also scan the height of the object from where it is placed.

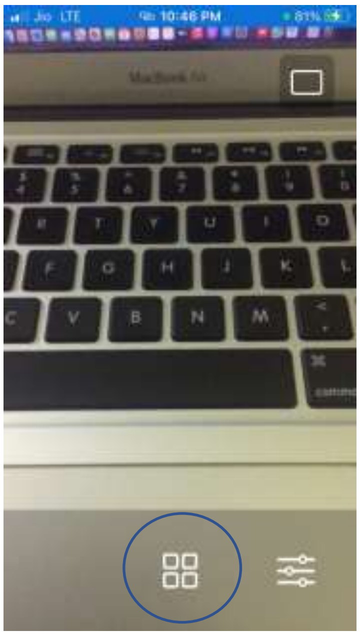
d. Multiuser:

Multiuser option in the is just visible in this version, its function will be updated in the next version of the system.

Functions of Buttons:

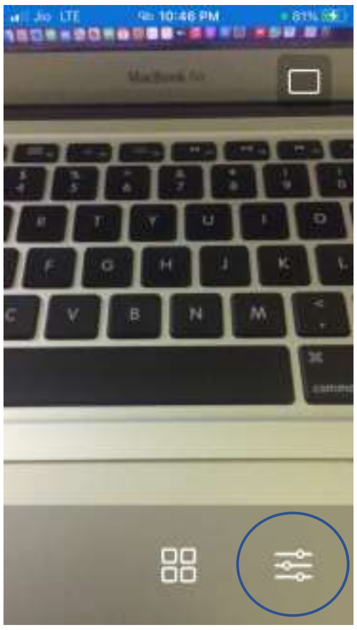
a. Browse Button:

Clicking on Browse button will open the Browser Panel with the multiple 3D objects to select.



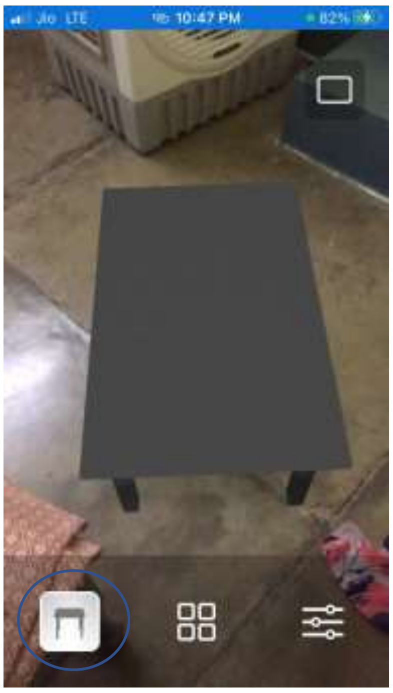
b. Setting Button:

Clicking on Setting button will open the Settings View with the additional features



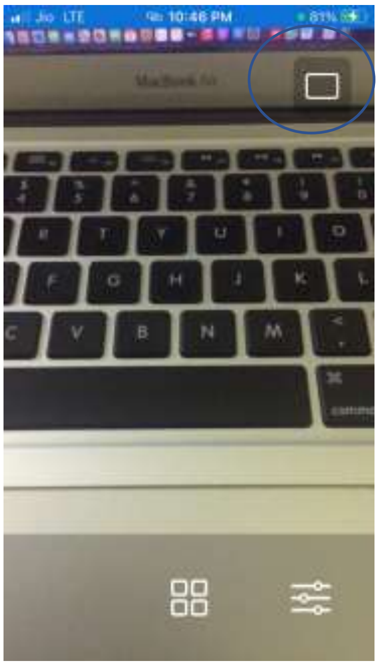
c. Most Recently Placed Button:

Clicking on the Most Recently Placed Button, will give shortcut to view last used 3D modal (object) to place in scanned view, this button allows the user to quickly place multiple instances of the same object without having to go to the browse view each time.



d. ControlVisibilityToggle Button:

This button allows the user to toggle the visibility of a ControlButtonBar.



5. Requirement Analysis

Hardware Requirements: -

- Ram – 8 GB or Above
- Processor (Name & Generation) – Core i5 or above, 5th Gen or Above
- Hard Disk – 512 GB HDD or Above / 128 SSD or Above
- Graphics – Intel Integrated HD 6000 or Above
- OS System – MacOS Big Sur

Hardware Used: -

1) **For writing and compiling code we used: -**

Figure 2:APPLE MacBook Air

Processor And Memory Features: -

Processor Brand	• Intel
Processor Name	• Core i5
Processor Generation	• 5th Gen
SSD Capacity	• 128 GB
RAM	• 8 GB
RAM Type	• DDR3
RAM Frequency	• 1600 MHz
Graphic Processor	• Intel Integrated HD 6000

Operating System: -

Operating System	• Mac OS Big Sur

2) **For testing the app, we used: -**

Figure 3:The first-generation iPhone SE

Specifications: -

RAM	• 2 GB
Internal Storage	• 128 GB
Camera (Rear / Front)	• 12MP Rear / 7MP Front

OS & Processor Features: -

Operating System	• iOS 14.2
Processor Type	• A13 Bionic Chip with 3rd Gen Neural Engine

Software Used: -

Software we used to write the Code: -

Xcode

XCode is Apple's integrated development environment (IDE) for macOS, used to develop software for macOS, iOS, iPadOS, watchOS, and tvOS. It was first released in 2003; the latest stable release is version 12.5, released on April 26, 2021, and is available via the Mac App Store free of charge for macOS Big Sur users. Registered developers can download preview releases and prior versions of the suite through the Apple Developer website. XCode includes Command Line Tools (CLT), which enable UNIX-style development via the Terminal app in macOS. They can also be downloaded and installed without the main IDE.

XCode supports source code for the programming languages C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez), and Swift, with a variety of programming models, including but not limited to Cocoa, Carbon, and Java. Third parties have added support for GNU Pascal, Free Pascal, Ada, C#, Go, Perl, and D.

Programming Language Used: -

Programming Language we used to write the Code: -

SwiftUI

SwiftUI is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. and the open-source community. First released in 2014, Swift was developed as a replacement for Apple's earlier programming language Objective-C, as Objective-C had been largely unchanged since the early 1980s and lacked modern language features. SwiftUI works with Apple's Cocoa and Cocoa Touch frameworks, and a key aspect of Swift's design was the ability to interoperate with the huge body of existing Objective-C code developed for Apple products over the previous decades. It is built with the open source LLVM compiler framework and has been included in XCode since version 6, released in 2014. On Apple platforms, it uses the Objective-C runtime library which allows C, Objective-C, C++ and Swift code to run within one program.

The features of SwiftUI are designed to work together to create a language that is powerful, yet fun to use. Some additional features of Swift include:

- Closures unified with function pointers
- Tuples and multiple return values
- Generics
- Fast and concise iteration over a range or collection
- Structs that support methods, extensions, and protocols
- Functional programming patterns, e.g., map and filter
- Powerful error handling built-in
- Advanced control flow with do, guard, defer, and repeat keywords

File Format & 3D Object Used: -

File Format we used for objects: -

. USDZ

USDZ is a 3D file format that displays 3D and AR content on iOS devices without having to download special apps. Users can easily share this portable format, and developers can exchange it between applications in their 3D-creation pipeline.

3D Object we used: -

Figure 5:3D Table

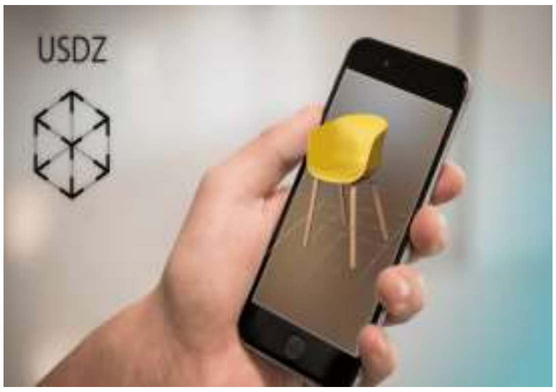


Figure 4: 3D Chair

6. Coding

A) BrowseView.swift

```
import SwiftUI

struct BrowseView: View {

    @Binding var showBrowse : Bool
    var body: some View{
        NavigationView{
            ScrollView(showsIndicators: false) {
//                 GridView for Thumbnails
                RecentsGrid(showBrowse: $showBrowse)

                ModelsByCategoryGrid(showBrowse: $showBrowse)
            }
            .navigationBarTitle(Text("Browse"),displayMode: .large)
            .navigationBarItems(trailing:
                                    Button(action: {
                                        self.showBrowse.toggle()
                                    }){
                                        Text("Done").bold()
                                    }
            )
        }
    }
}

struct RecentsGrid: View {
    @EnvironmentObject var placementSettings: PlacementSettings
    @Binding var showBrowse : Bool

    var body: some View {
        if !self.placementSettings.recentlyPlaced.isEmpty {
            HorizontalGrid(showBrowse: $showBrowse, title: "Recents", items:
getRecentsUniqueOrdered())
        }
    }

    func getRecentsUniqueOrdered() -> [Model] {
        var recentsUniqueOrderedArray: [Model] = []
        var modelNameSet: Set<String> = []

        for model in self.placementSettings.recentlyPlaced.reversed() {

            if !modelNameSet.contains(model.name) {
                recentsUniqueOrderedArray.append(model)
                modelNameSet.insert(model.name)
            }
        }

        return recentsUniqueOrderedArray
    }
}

struct ModelsByCategoryGrid: View{
    @Binding var showBrowse : Bool
    let models = Models()
    var body: some View{
        VStack{
            ForEach(ModelCategory.allCases, id: \.self){ category in

                //Only display grid if category contains items
                if let modelsByCategory = models.get(category: category){
                    HorizontalGrid(showBrowse:      $showBrowse,      title:
category.label, items: modelsByCategory)
                }
            }
        }
    }
}

struct HorizontalGrid : View {
    @EnvironmentObject var placementSettings : PlacementSettings

    @Binding var showBrowse : Bool
    var title: String
    var items: [Model]

    private let gridItemLayout = [GridItem(.fixed(150))]

    var body: some View{
        VStack(alignment: .leading) {
            Separator()

            Text(title)
                .font(.title2.bold())
                .padding(.leading,20)
                .padding(.top,20)

            ScrollView(.horizontal, showsIndicators : false){

                LazyHGrid(rows: gridItemLayout, spacing:30){
                    ForEach(0..<items.count) { index in

//                          Color(UIColor.secondarySystemFill)
//                              .frame(width: 150, height: 150)
//                              .cornerRadius(8)
                        let model =  items[index]

                        ItemButtom(model: model){

                            model.asyncLoadModelEntity()

                            self.placementSettings.selectedModel = model

                            print("BrowserView:   selected\(model.name)   for
placement.")
                            self.showBrowse = false
                        }

                    }
                }
                .padding(.horizontal,22)
                .padding(.vertical , 10)

            }
        }
    }
}

struct  ItemButtom : View {
    let model : Model
    let action: () -> Void

    var body: some View{
        Button(action:{
            self.action()
        }){
            Image(uiImage: self.model.thumbnail)
                .resizable()
                .frame(height: 150)
                .aspectRatio(1/1 ,contentMode: .fit)
                .background(Color(UIColor.secondarySystemFill))
                .cornerRadius(8.0)

        }
    }
}

struct  Separator:View {
    var body: some View{
        Divider()
            .padding(.horizontal,20)
            .padding(.vertical,10)
    }
}
```

B) ContentView.swift

```
import SwiftUI
import RealityKit

struct ContentView: View {

    @EnvironmentObject var placementSettings : PlacementSettings
    @State private var isControlVisible : Bool = true
    @State private var showBrowse : Bool = false
    @State private var showSettings : Bool = false


    var body: some View {
        ZStack(alignment: .bottom){
            ARViewContainer()

            if self.placementSettings.selectedModel == nil {
                ControlView(isControlVisible: $isControlVisible, showBrowse: 
$showBrowse ,showSettings : $showSettings )
            }else{
                PlacementView()
            }

        }
        .edgesIgnoringSafeArea(.all)

    }
}
```

```
struct ARViewContainer :
    UIViewRepresentable {

    @EnvironmentObject var placementSettings : PlacementSettings
    @EnvironmentObject var sessionSettings : SessionSettings

    func makeUIView(context: Context) ->
    CustomeARView {
        let   arView   =   CustomeARView(frame:   .zero,   sessionSettings:
sessionSettings)

//        subscriber to showevents
        self.placementSettings.sceneObserver    =  arView.scene.subscribe(to:
SceneEvents.Update.self, { (event) in

            self.updateScene(for: arView)
        })
        return arView
    }
    func updateUIView(_ uiView: CustomeARView, context: Context) {}

    //add new func in ARView
    private func updateScene(for arView: CustomeARView){
        //Only display focusEntity when the user has selected a model for
placement
        arView.focusEntity?.isEnabled = self.placementSettings.selectedModel
!= nil

        //Add model to scene if confirmed for placement
        if let confirmedModel = self.placementSettings.confirmedModel, let
modelEntity = confirmedModel.modelEntity{

            self.place(modelEntity, in: arView)

            self.placementSettings.confirmedModel = nil
        }
    }

    //after updateScene add place() function
    private func place(_ modelEntity: ModelEntity, in arView: ARView){
        //1. Clone modelEntity. This creates an identical copy of modelEntity
and reference the same model. This also allows us to have multiple models of
the same asset is our scene.

        let clonedEntity = modelEntity.clone(recursive: true)


        //2.Enable translation and rotation gestures.

        clonedEntity.generateCollisionShapes(recursive: true)
        arView.installGestures([.translation, .rotation], for: clonedEntity)

        //3. Created an anchorEntity and add clonedEntity to the anchorEntity.

        let anchorEntity = AnchorEntity(plane: .any)
        anchorEntity.addChild(clonedEntity)
        //4. Add the anchorEntity to the arView.scene
        arView.scene.addAnchor(anchorEntity)

        print("Added modelEntity to scene.")
    }


}
struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
            .environmentObject(PlacementSettings())
            .environmentObject(SessionSettings())

    }
}
```

C)ControlView.swift

```
import SwiftUI

struct ControlView:View {
    @Binding var isControlVisible : Bool
    @Binding var showBrowse : Bool
    @Binding var showSettings: Bool

    var body: some View{
        VStack{
            ControlVisibilityToggleButton(isControlVisible:
$isControlVisible)

            Spacer()

            if isControlVisible{
                ControlButtonBar(showBrowse:  $showBrowse  ,  showSettings  :
$showSettings)
            }

        }
    }
}

struct ControlVisibilityToggleButton:View {
    @Binding var isControlVisible : Bool
    var body: some View{
        HStack{
            Spacer()
            ZStack{
                Color.black.opacity(0.25)

                Button(action: {
                   print("Control Visibility Button Pressed.")
                    self.isControlVisible.toggle()
                }){
                    Image(systemName:  self.isControlVisible  ?  "rectangle"  :
"slider.horizontal.below.rectangle")
                        .font(.system(size: 25))
                        .foregroundColor(.white)
                        .buttonStyle(PlainButtonStyle())
                }
            }
            .frame(width: 50, height: 50)
            .cornerRadius(8.0)
        }
        .padding(.top,45)
        .padding(.trailing,20)
    }
}


struct ControlButtonBar:View {
    @EnvironmentObject var placementSettings : PlacementSettings
    @Binding var showBrowse : Bool
    @Binding var showSettings: Bool

    var body: some View{
        HStack{
//          Most Recently Placeed Button pressed
//             ControlButton(systemIconName: "clock.fill") {
//                 print("MostRecentlyPlaced Button Pressed")
//             }


MostRecentlyPlacedButton().hidden(self.placementSettings.recentlyPlaced.isEm
pty)

            Spacer()
//            Browse button pressed
            ControlButton(systemIconName:  "square.grid.2x2") {
                print("Browse Button Pressed")
                self.showBrowse.toggle()
            }.sheet(isPresented: $showBrowse, content: {
//                Browser View return
                BrowseView(showBrowse: $showBrowse)
            })

            Spacer()
//            Setting button pressed
            ControlButton(systemIconName: "slider.horizontal.3") {
                print("Setting Button Pressed")
                self.showSettings.toggle()
            }.sheet(isPresented: $showSettings) {
                SettingsView(showSettings: $showSettings)
            }



        }
        .frame(minWidth:200)
        .padding(30)
        .background(Color.black.opacity(0.25))
    }
}

struct ControlButton : View {
    let systemIconName : String
    let action: () -> Void
    var body: some View{

        Button(action: {
            self.action()
        }){
            Image(systemName: systemIconName)
                .font(.system(size: 35))
                .foregroundColor(.white)
                .buttonStyle(PlainButtonStyle())
        }
        .frame(width: 50, height: 50)
    }
}

// add new struct MostRecentlyPlacedButton

struct MostRecentlyPlacedButton: View{
    @EnvironmentObject var placementSettings : PlacementSettings

    var body: some View{
        Button(action: {
            print("Most Recently Placed button pressed.")
            self.placementSettings.selectedModel                              =
self.placementSettings.recentlyPlaced.last
        }) {
            if           let           mostRecentlyPlacedModel            =
self.placementSettings.recentlyPlaced.last{
                Image(uiImage: mostRecentlyPlacedModel.thumbnail)
                    .resizable()
                    .frame(width: 46)
                    .aspectRatio(1/1, contentMode: .fit)
            }else {
                Image(systemName: "clock.fill")
                    .font(.system(size: 35))
                    .foregroundColor(.white)
                    .buttonStyle(PlainButtonStyle())
            }
        }
        .frame(width: 50, height: 50)
        .background(Color.white)
        .cornerRadius(8.0)
    }
}
```

D)CustomARView.swift

```
import RealityKit
import ARKit
import FocusEntity
import SwiftUI
import Combine

class CustomeARView :  ARView{
    var focusEntity: FocusEntity?
    var   sessionSettings: SessionSettings

    private var peopleOcclusionCancellable: AnyCancellable?
    private var objectOcclusionCancellable: AnyCancellable?
    private var lidarDebugCancellable: AnyCancellable?
    private var multiuserCancellable: AnyCancellable?

    required   init(frame   frameRect:   CGRect   ,   sessionSettings   :
SessionSettings){

        self.sessionSettings = sessionSettings

        super.init(frame: frameRect)

        focusEntity = FocusEntity(on: self, focus: .classic)

        configure()

        self.initializeSettings()

        self.setupSubscribers()
    }

    @objc required dynamic init?(coder decoder: NSCoder){
        fatalError("init(coder:) has not been implemented")
    }

    @objc required dynamic init(frame frameRect: CGRect) {
        fatalError("init(frame:) has not been implemented")
    }

    private func configure(){
        let config = ARWorldTrackingConfiguration()
        config.planeDetection = [.horizontal, .vertical]

        if ARWorldTrackingConfiguration.supportsSceneReconstruction(.mesh) {
            config.sceneReconstruction = .mesh
        }

        session.run(config)
    }

    private func initializeSettings() {
        self.updatePeopleOcclusion(isEnabled:
sessionSettings.isPeopleOcclusionEnabled)

        self.updateObjectOcclusion(isEnabled:
sessionSettings.isObjectOcclusionEnabled)

        self.updateLidarDebug(isEnabled:
sessionSettings.isLidarDebugEnabled)

        self.updateMultiuser(isEnabled: sessionSettings.isMultiuserEnabled)
    }

    private func setupSubscribers(){
        self.peopleOcclusionCancellable                                     =
sessionSettings.$isPeopleOcclusionEnabled.sink{ [weak self]
            isEnabled in
            self?.updatePeopleOcclusion(isEnabled: isEnabled)
        }
        self.objectOcclusionCancellable                                     =
sessionSettings.$isObjectOcclusionEnabled.sink { [weak self] isEnabled in
            self?.updateObjectOcclusion(isEnabled: isEnabled)
        }

        self.lidarDebugCancellable                                    =
sessionSettings.$isLidarDebugEnabled.sink {[weak self] isEnabled in
            self?.updateLidarDebug(isEnabled: isEnabled)
        }

        self.multiuserCancellable = sessionSettings.$isMultiuserEnabled.sink
{[weak self] isEnabled in
            self?.updateMultiuser(isEnabled: isEnabled)
        }
    }
    private func updatePeopleOcclusion(isEnabled: Bool){
        print("\(#file): isPeopleOcclusionEnabled is now \(isEnabled)")

        guard
ARWorldTrackingConfiguration.supportsFrameSemantics(.personSegmentationWithD
epth) else {
            return
        }

        guard   let   configuration   =   self.session.configuration   as?
ARWorldTrackingConfiguration else {
            return
        }

        if
configuration.frameSemantics.contains(.personSegmentationWithDepth) {

configuration.frameSemantics.remove(.personSegmentationWithDepth)
        }else {

configuration.frameSemantics.insert(.personSegmentationWithDepth)
        }

        self.session.run(configuration)
    }
    private func updateObjectOcclusion(isEnabled: Bool) {
        print("\(#file): isObjectOcclusionEnabled is now \(isEnabled)")

        if self.environment.sceneUnderstanding.options.contains(.occlusion) {
            self.environment.sceneUnderstanding.options.remove(.occlusion)
        } else {
            self.environment.sceneUnderstanding.options.insert(.occlusion)
        }
    }
    private func updateLidarDebug(isEnabled: Bool) {
        print("\(#file): isLidarDebugEnabled is now \(isEnabled)")

        if self.debugOptions.contains(.showSceneUnderstanding) {
            self.debugOptions.remove(.showSceneUnderstanding)
        } else {
            self.debugOptions.insert(.showSceneUnderstanding)
        }
    }
    private func updateMultiuser(isEnabled: Bool) {
        print("\(#file): isMultiuserEnabled is now \(isEnabled)")
    }
}
```

E) Model.swift

```
import SwiftUI
import RealityKit
import Combine

enum ModelCategory: CaseIterable{
    case table
    case chair
    case decor
    case light

    var label: String{
        get{
            switch self{
            case .table:
                return "Tables"
            case .chair:
                return "Chairs"
            case .decor:
                return "Decor"
            case .light:
                return "Lights"
            }
        }
    }
}


class Model{
    var name: String
    var category: ModelCategory
    var thumbnail: UIImage
    var modelEntity: ModelEntity?
    var scaleCompensation: Float

    private var cancellable: AnyCancellable?


    init(name: String,category: ModelCategory,scaleCompensation: Float=0.1){
        self.name=name
        self.category=category
        self.thumbnail=UIImage(named:name) ?? UIImage(systemName:"photo")!
        self.scaleCompensation=scaleCompensation
    }


    func asyncLoadModelEntity(){
        let filename=self.name  + ".usdz"

        self.cancellable = ModelEntity.loadModelAsync(named:filename)
            .sink(receiveCompletion: { loadCompletion in

                switch loadCompletion{
                    case.failure(let error):print("Unable to load modelEntity 
for \(filename).Error: \(error.localizedDescription)")
                    case.finished:
                        break
                }
            },receiveValue: {modelEntity in
```

```
                self.modelEntity = modelEntity
                self.modelEntity?.scale*=self.scaleCompensation

                print("modelEntity for \(self.name) has been loaded.")
            })
    }
}


struct Models{
    var all: [Model] = []

    init(){
        //Table
        let Table = Model(name:"table", category: .table, scaleCompensation:
0.32/100)
        let  ComputerTable  =  Model(name:"computerTable",  category:  .table,
scaleCompensation: 0.32/100)

        self.all += [Table,ComputerTable]

        //Chairs
      //   let  diningChair  =  Model(name:"dinning_chair",  category:  .chair,
scaleCompensation:0.32/100)
        let   eamesChairBlack   =   Model(name:"chair",   category:   .chair,
scaleCompensation:0.32/100)
        let   blackChair   =   Model(name:   "chairBlack",   category:   .chair,
scaleCompensation: 0.32/100)
        let    oakChair    =    Model(name:    "oakChair",    category:    .chair,
scaleCompensation: 0.32/100)
        let  MammothChair  =  Model(name:  "MammothChair",  category:  .chair,
scaleCompensation: 0.32/100)

        self.all += [eamesChairBlack,blackChair,oakChair,MammothChair]

        //Decor
        let   flower   =   Model(name:   "flower_tulip",   category:   .decor,
scaleCompensation: 5/100)
        let  cupandsaucer  =  Model(name:  "cupandsaucer",  category:  .decor,
scaleCompensation: 0.32/100)
        let    teapot    =    Model(name:    "teapot",    category:    .decor,
scaleCompensation: 0.32/100)
        let GrenadaCommode = Model(name: "GrenadaCommode", category: .decor,
scaleCompensation: 0.32/100)

        self.all += [flower,cupandsaucer,teapot,GrenadaCommode]

        //Lights
        let    plane    =    Model(name:    "biplane",    category:    .light,
scaleCompensation: 0.32/100)

        self.all += [plane]

//         chairBlack


    }
    func get(category : ModelCategory)-> [Model]{
        return all.filter( {$0.category == category})
    }
}
```

F) PlacementSettings.swift

```
import SwiftUI
import RealityKit
import Combine

class PlacementSettings : ObservableObject {


    //When the user selectes a model in BrowseView, this property is set.
    @Published var selectedModel: Model?{
        willSetalue){
            print("Setting     selectedModel     to     \(String(describing:
newValue?.name))")
        }
    }
    (newValue
    //When the user taps confirm in PlacementView, the value of selectedModel
is assigned to confirmedModel
    @Published var confirmedModel: Model?{
        willSet(newValue){
            guard let model = newValue else{
                print("Clearing confirmedModel")
                return
            }

            print("Setting confirmedMOdel to \(model.name)")

            self.recentlyPlaced.append(model)

        }
    }

    //This property retains a record of placed models in the scene. The last
element in the array is the most recently placed model.
    @Published var recentlyPlaced: [Model] = []


//    this property returns the cancellable object for scene
    var sceneObserver : Cancellable?
}
```

G) PlacementView.swift

```
import SwiftUI

struct PlacementView: View{
    @EnvironmentObject var placementSettings : PlacementSettings

    var body: some View{
        HStack{

            Spacer()

            PlacementButton(systemIconName: "xmark.circle.fill"){
                print("Cancel Placement button pressed.")
                self.placementSettings.selectedModel = nil
            }

            Spacer()

            PlacementButton(systemIconName: "checkmark.circle.fill"){
                print("Confirm Placement button pressed.")

                self.placementSettings.confirmedModel                       =
self.placementSettings.selectedModel

                self.placementSettings.selectedModel = nil
            }

            Spacer()
        }
        .padding(.bottom,30)
    }
}

struct PlacementButton: View{
    let systemIconName: String
    let action: () -> Void

    var body: some View{
        Button(action:{
            self.action()
        }) {
            Image(systemName: systemIconName)
                .font(.system(size: 50,weight: .light,design: .default))
                .foregroundColor(.white)
                .buttonStyle(PlainButtonStyle())
        }
        .frame(width: 75, height: 75)
    }
}
```

H) SessionSetting.swift

```
import SwiftUI

class SessionSettings: ObservableObject {

    @Published var isPeopleOcclusionEnabled: Bool = false
    @Published var isObjectOcclusionEnabled: Bool = false
    @Published var isLidarDebugEnabled: Bool = false
    @Published var isMultiuserEnabled: Bool = false
}
```

I)SettingsView.swift

```
import SwiftUI

enum Setting {
    case peopleOcclusion
    case objectOcclusion
    case lidarDebug
    case multiuser

    var lable: String {
        get{
            switch self{
```

```
            case .peopleOcclusion, .objectOcclusion:
                return "Occlucion"
            case .lidarDebug:
                return "LiDAR"
            case .multiuser:
                return "Multiuser"
            }
        }
    }

    var systemIconName: String{
        get{
            switch self{
            case .peopleOcclusion:
                return "person"
            case .objectOcclusion:
                return "cube.box.fill"
            case .lidarDebug:
                return "light.min"
            case .multiuser:
                return "person.2"
            }
        }
    }
}

struct SettingsView: View{
    @Binding var showSettings: Bool

    var body: some View{
        NavigationView{
            SettingsGrid()
                .navigationBarTitle(Text("Settings"), displayMode: .inline)
                .navigationBarItems(trailing:
                    Button(action: {
                        self.showSettings.toggle()
                    }) {
                        Text("Done").bold()
                    })
        }
    }
}

struct SettingsGrid: View {
    @EnvironmentObject var sessionSettings: SessionSettings

    private var gridItemLayout = [GridItem(.adaptive(minimum: 100, maximum:
100),spacing: 25)]

    var body: some View{
        ScrollView {

            LazyVGrid(columns: gridItemLayout, spacing: 25){
                    SettingToggleButton(setting:   .peopleOcclusion,   isOn:
$sessionSettings.isPeopleOcclusionEnabled)

                    SettingToggleButton(setting:   .objectOcclusion,   isOn:
$sessionSettings.isObjectOcclusionEnabled)
```

```
                    SettingToggleButton(setting:      .lidarDebug,      isOn:
$sessionSettings.isLidarDebugEnabled)

                    SettingToggleButton(setting:      .multiuser,       isOn:
$sessionSettings.isMultiuserEnabled)
            }
        }
        .padding(.top, 35)
    }
}

struct SettingToggleButton: View {
    let setting: Setting
    @Binding var isOn: Bool

    var body: some View{
        Button(action: {
            self.isOn.toggle()
            print("\(#file) - \(setting): \(self.isOn)")
         }) {
            VStack{

                Image(systemName: setting.systemIconName)
                        .font(.system(size: 35))
                    .foregroundColor(self.isOn  ?  .green  :  Color(UIColor
.secondaryLabel ))
                        .buttonStyle(PlainButtonStyle())

                Text(setting.lable)
                    .font(.system(size:   17,   weight:   .medium,   design:
.default))
                    .foregroundColor(self.isOn  ?  Color(UIColor  .label)  :
Color(UIColor .secondaryLabel))
                    .padding(.top,5)
            }
        }
        .frame(width: 100, height: 100)
        .background(Color(UIColor.secondarySystemFill))
        .cornerRadius(20.0)
    }
}
```

J)StyleMySpace3D.swift

```
import SwiftUI

@main
struct SMS3App: App {
    @StateObject var placementSettings  = PlacementSettings()
    @StateObject var sessionSettings = SessionSettings()
    var body: some Scene {
        WindowGroup {
            ContentView()
                .environmentObject(placementSettings)
                .environmentObject(sessionSettings)

        }
    }
}
```

K) View+Extensions.swift

```
import SwiftUI

extension View{
        @ViewBuilder func hidden(_ shouldHide: Bool) -> some View{
            switch shouldHide{
            case true: self.hidden()
            case false: self
            }
        }
}
```

7. Advantages And Disadvantages

➢ **Advantages: -**

1. This helps the client visualize the project before it is developed. It gives an ability to user to design the space the way they want.
2. User will also get an ability to edit the design and make changes even if the design is at the final stage. So, now users need not worry about the tedious corrections that done in actual decor and furniture.
3. AR in system will let user guide the designer in the best possible manner Even the minute details related to the designing process can be communicated interactively.
4. The client can try various products for their new project such as the furniture without paying a penny.
5. Using this system, it would be much easier for designer to present their innovative ideas to client.
6. The designers can gain a competitive edge by giving their clients a complete view and information about their future project.

➢ **Disadvantages: -**

1. It is quite expensive to use AR based system and maintain it. Moreover, it is costly to use it in daily life.
2. For designing using system requires training and specific tools to be made available to designers.
3. One major drawback of AR based application is lack of privacy.

8. Future Scope

In our increasingly digitized world, where everything from our phones to our thermostats is “smart,” it should come as no surprise that virtual reality (VR) and augmented reality (AR) are starting to play a role in the design industry. It’s a logical relationship: designers have always imagined new or re-imagined existing spaces, created a drawing via analog means or digital ones, and presented that visualization to their clients. It tracks, then, that as drawing technology has advanced from paper, onto a screen, and finally into a three-dimensional projection, designers have adapted to the times. Designers are used to interpreting 2D representations of designs like floor plans, elevations, and finish boards, non-design professionals typically have more trouble visualizing the experience of being in spaces proposed by 2D representations,” says Jess Bayuk, a design technology specialist in global design firm HOK’s New York office. “Virtual reality and augmented reality allow viewers to get a better sense of how a space will look and feel in a 3D mode of representation that is more like the everyday experience.”

Most people find it difficult to renovate a house or to choose furniture because they cannot see what the final result will look like. This applies both to those who adopt a do-it-yourself approach and to those who employ professional designers. Here, therefore, Augmented Reality applications can come to the aid of the various phases of interior design. Architects, designers and real estate marketing experts are the first who can take advantage of these technologies: by realistically visualizing how the imagined projects will look, they have the possibility to easily design the environments and to show them to the end customer with greater effectiveness. When in the interior design phase, the attention goes from the functionality of the various elements to the aesthetic aspects, errors of interpretation can occur. Thanks to Augmented Reality, however, we are all able to experiment with all the options - furniture, lighting, flooring, paint on the walls, types of fixtures, etc. - and define the appearance of the rooms before putting our hand to the wallet.

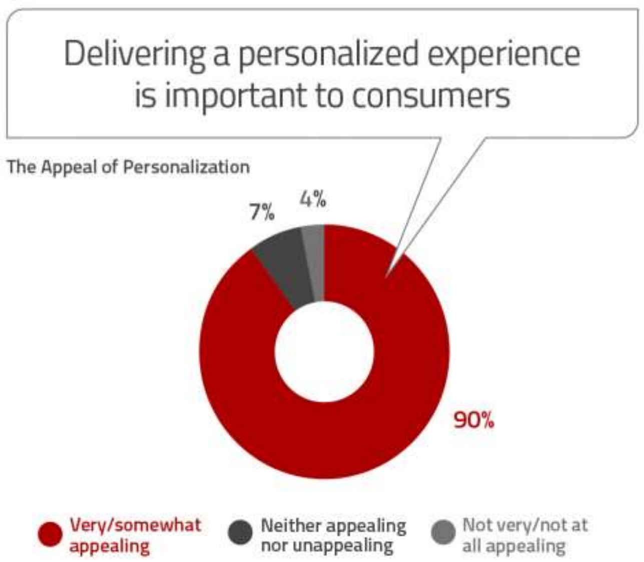


Figure 6:The Appeal of Personalization

3D augmented reality and Virtual reality are the technologies that are serving the needs of new-age customers. Personalization appeals 90% of the customers hence using AR, designers, and architects can attract and retain the customers like never before.

Augmented and Virtual reality is the future of interior design. Things that were once impossible are now possible with these immersive technologies. Printed catalogs may get obsolete making interior designing more intelligent, convenient and lively.

9. Conclusion

A new age customer wants a customized or personalized solution exactly according to their needs. Augmented reality used in the interior design is the revolutionary solution to the constraints faced by clients and designers in visualizing the actual idea like never before. It is easier to fit into our daily lives due to the utilization of mobile phones. 3D models of furniture capture the best possible details and realism that enables customers to see the object from every angle to get how it fits, this eases the overall experience of the users and lets the designers to promote System and the user see furniture or wall paint that would look real! The system implies customer can preciously check whether the object will fit in the room or not before making the purchase, user will also get an ability to edit the designs and make changes even if the design is at the final stage, that’s how user need not worry about the tedious corrections that are done in the actual decor and furniture. This system will be time saving for the both customer and designer and also it will be easy for the designer to clear the idea to the client. Things that were once impossible are now possible with this immersive technology. Because of this may get obsolete making interior designing more intelligent, convenient and lively. This leads to a more informed buying experience in every sense. This project proposes a solution to design and build an application that will run on smartphones. The designers can gain a competitive edge by giving their clients a complete view and information about their future project.

References

- https://developer.swift.com/
- https://sketchfab.com/tags/usdz
- https://developer.apple.com/augmented-reality/quick-look/
- https://youtube.com/playlist?list=PLBv1NzmBcY51F-pdOIywpndccaB21NTid