

A Project Report on

*“ARchiTECH”*



Submitted by

*Jash Seth - 1698024*

*Abhijit Turate – 1698038*

*Kunal Vishwasrao - 1698041*

Guided By

*Mr. Abhijit Dongaonkar*

Term – December’18 to April’19

Information Technology Department

SVKM’s

Shri Bhagubhai Mafatlal Polytechnic,

Irla, N. R. G. Marg, Vile Parle (W), Mumbai – 400056

****

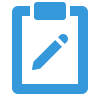
ACKNOWLEDGMENTS

**Acknowledgements**

* We take this opportunity to express our profound gratitude and deep regards to our guide Mr. Abhjit Dongaonkar (Head of Department, IT, SBMP) for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.
* The blessings, help and guidance by him time to time shall carry us a long way in the journey of life on which we are about to embark.
* We also take this opportunity to express a deep sense of gratitude to Mrs. Neeta Kadukar (Vice Principal, SBMP) for her coordinal support, valuable information and guidance which helped us in completing this task through various stages.
* We are obliged to staff members and technicians of IT department for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our project.
* Lastly, we thank almighty, our parents and our classmates for their constant encouragement without which this project would not have been possible.

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Table of Contents** | **Page No.** |
| **1.** | **Abstract** | **6** |
| **2.** | **Introduction** | **8** |
| **3.** | **Analysis** | **10** |
| **4.** | **Augmented Reality** | **12** |
| **5.**  5.1  5.2  5.3  5.4  5.5  5.6  5.7 | **Technology Used**  Unity  Why Unity?  Scripting in Unity  Vuforia  Visual Studio  C#  Why C#? | **14**  15  16  17  19  20  21  23 |
| **6** | **Requirements** | **24** |
| **7** | **Case Study** | **26** |
| **8** | **Use Case Diagram** | **29** |
| **9** | **Flowchart** | **31** |
| **10** | **Activity Diagram** | **33** |
| **11** | **Source Code** | **35** |
| **12** | **Screenshots** | **42** |
| **13** | **Advantages** | **46** |
| **14** | **Limitations** | **48** |
| **15** | **Future Scope** | **50** |
| **16** | **Conclusion** | **52** |
| **17** | **References** | **54** |



ABSTRACT

**Abstract**

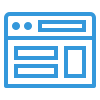
As we know, the first step for building anything is to create a model of the structure of the object to be built. The model acts as the first prototype of the object.

A model provides a rough idea about the structure of the actual object and acts as guideline for constructing the object.

Architects create a 2D model of a given structure, which is to be built or does already exist. However, 2D models does not provide a detailed view of the actual structure.

The common people also find it difficult to interpret and analyse the structure portrayed in 2D model. A 3D model for the given structure would be more understandable and more appealing to the people. Hence, it would be better to create 3D models of the structures.

But 3D models require special skill set to create, and takes more time. Thus, a technology to convert the already existing 2D models into 3D view model is required.

****

INTRODUCTION

**Introduction**

Augmented reality is the integration of digital information with the user's environment in real time. Unlike virtual reality, which creates a totally artificial environment, augmented reality uses the existing environment and overlays new information on top of it. Today, Google glass and heads-up displays in car windshields are perhaps the most well-known consumer AR products, but the technology is used in many industries including healthcare, public safety, gas and oil, tourism and marketing. AR is a very famous and rapidly growing field. It is used almost all the fields for better understanding of a concept.

ARchiTECH is such an app that helps common people understand the 2D floor plans drawn by architects by creating a 3D model of that given plan just by scanning it. Understanding 3D models is way easier than understanding 2D floor plans drawn by an architect using some notations.

The application is capable of creating a 3D model of a given 2D plan. It first scans the given plan and then builds a corresponding 3D model by replacing the lines forming the rooms with actual walls. This process requires a 3D game engine to build the application; which in this case is Unity.



Analysis

**Time Management**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Weeks | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | | 15 |
| Information  Gathering |  | |  | | | | | | | | | | | | | |
| Designing |  | |  | | | | |  | | | | | | | | |
| Coding |  | | | | | | |  | | | |  | | | | |
| Implementation |  | | | | | | | | | | |  | | |  | |
| Testing |  | | | | | | | | | | | | | |  | |



**Augmented Reality**

**What is Augmented Reality?**

Augmented reality (AR) is an interactive experience of a real-world environment where the objects that reside in the real-world are "augmented" by computer-generated perceptual information, sometimes across multiple sensory modalities, including visual, auditory, haptic, somatosensory, and olfactory. Augmented reality is related to two largely synonymous terms: mixed reality and computer-mediated reality.



Augmented reality is used to enhance natural environments or situations and offer perceptually enriched experiences. With the help of advanced AR technologies (e.g. adding computer vision and object recognition) the information about the surrounding real world of the user becomes interactive and digitally manipulatable. Information about the environment and its objects is overlaid on the real world. This information can be virtual or real, e.g. seeing other real sensed or measured information such as electromagnetic radio waves overlaid in exact alignment with where they actually are in space.



TECHNOLOGY USED

1. **Unity**

Unity is a cross-platform real-time engine developed by Unity Technologies, first announced and released in June 2005 at Apple Inc.'s Worldwide Developers Conference as an OS-X-exclusive game engine. As of 2018, the engine has been extended to support 27 platforms. The engine can be used to create both three-dimensional and two-dimensional games as well as simulations for its many platforms. Several major versions of Unity have been released since its launch, with the latest stable version being Unity 2018.3.12.



Unity3D is a commercially available multiplatform game engine used for the production of 2d and 3D video games as well as non-game interactive simulations and visualizations.

Unity gives users the ability to create games and interactive experiences in both 2D and 3D, and the engine offers a primary scripting API in C#, for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality.

Unity is one of the most popular game engines available due to its combination of power, flexibility, and ease of use.

1. **Why Unity?**

The Unity editor is supported on Windows and macOS, with a version of the editor available for the Linux platform, albeit in an experimental stage, while the engine itself currently supports building games for 29 different platforms. The platforms are listed as following: iOS, Android, Tizen, Windows (Vista or newer), Universal Windows Platform, macOS, Linux, WebGL, PlayStation 4, PlayStation Vita, Xbox One, Wii U, 3DS, Oculus Rift, Google Cardboard, SteamVR, PlayStation VR, Gear VR, Windows Mixed Reality, Daydream, Android TV, Samsung Smart TV, tvOS, Nintendo Switch, Fire OS, Facebook Gameroom, Apple's ARKit, Google's ARCore, and Vuforia.

Unity Pro is available for a fee, and Unity Personal has no fee; it is available for any use to individuals or companies with less than US$100,000 of annual gross revenue. On March 3, 2015 Unity Technologies made available the complete engine with their upcoming new version 5 (Unity 5) for free including all features, less source code and support.

Unity is the most widely used VR development platform, and over 91% of HoloLens experiences are Made with Unity. Whether it's VR, AR, or MR, you can count on Unity's highly optimized rendering pipeline and the rapid iteration capabilities of our Editor to make your XR creative vision a reality.

##### **Scripting in Unity**

##### **What is scripting in Unity?**

Scripting tells our Game Objects how to behave; it’s the scripts and components attached to the Game Objects, and how they interact with each other, that creates your gameplay. Now, scripting in Unity is different from pure programming. If you’ve done some pure programming, e.g. you created a running app, you should realize that in Unity you don’t need to create the code that runs the application, because Unity does it for you. Instead, you focus on the gameplay in your scripts.

Unity runs in a big loop. It reads all of the data that’s in a game scene. For example, it reads through the lights, the meshes, what the behaviors are, and it processes all of this information for you.

If you think about television, where, for example in North America, you have 29.5 frame/sec, Unity needs to do the same thing. It’s running single discrete frames, one after another. You direct Unity with the instructions that you write in your scripts, and Unity executes them frame after frame as fast as it can.

Achieving a high frame rate means not only your game will look more fluid, but your scripts will also be executed more often, making controls more responsive.

##### **What languages can you use in Unity?**

A script must be attached to a Game Object in the scene in order to be called by Unity. Scripts are written in a special language that Unity can understand. And, it’s through this language that we can talk to the engine and give it our instructions.

The language that’s used in Unity is called C# (pronounced C-sharp). All the languages that Unity operates with are object-oriented scripting languages. Like any language, scripting languages have syntax, or parts of speech, and the primary parts are called variables, functions, and classes.

If you’re using a version of Unity until 2017.3, you’ll notice that it has a text editor called Mono Develop: it can help us complete our code, it’ll let us know if we’re writing a wrong piece of code, and allows us to take shortcuts. Starting with 2018.1, you can also use Visual Studio for Unity Community, or other text editors such as Visual Studio, Notepad, or Sublime text.

##### **What do these do?**

**Variables** hold values and references to objects (you can see objects as “bigger” variables). They’re like a box that holds something for us to use. Variables start with a lowercase letter.

**Functions** are collections of code that compare and manipulate these variables. Functions start with an uppercase letter. We organize code in functions so that they can be easily reused multiple times in different parts of the program.

**Classes** are a way to structure code to wrap collections of variables and functions together to create a template that defines the properties of an object.

Scripting is primarily comparing these objects and their current states and values. It’s based on logic determining an outcome or resolution.

1. **Vuforia**

Vuforia is an augmented reality software development kit (SDK) for mobile devices that enables the creation of augmented reality applications. It uses computer vision technology to recognize and track planar images (Image Targets) and simple 3D objects, such as boxes, in real time.



This image registration capability enables developers to position and orient virtual objects, such as 3D models and other media, in relation to real world images when they are viewed through the camera of a mobile device.

Vuforia provides Application Programming Interfaces (API) in C++, Java, Objective-C++ (a language utilizing a combination of C++ and Objective-C syntax), and the .NET languages through an extension to the Unity game engine.

In this way, the SDK supports both native development for iOS and Android while it also enables the development of AR applications in Unity that are easily portable to both platforms. AR applications developed using Vuforia are therefore compatible with a broad range of mobile devices including the iPhone, iPad, and Android phones and tablets running Android OS version 2.2 or greater and an ARMv6 or 7 processor with FPU (Floating Point Unit) processing capabilities.

1. **Visual Studio**

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight.



Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS. Support for other languages such as Python, Ruby, Node.js, and M among others is available via plug-ins.

The most basic edition of Visual Studio, the Community edition, is available free of charge. The currently supported Visual Studio version is 2019.

**Features:**

* Code Editor
* Debugger
* Designer
* Extensibility
* Other Tools

1. **C#**

The programming language used in unity for developing this game is C# also called as C Sharp.

C# (pronounced as see sharp) is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component- oriented programming languages designed for the Common Language Infrastructure.

C# is intended to be a simple, modern, general-purpose, object-oriented programming language. The most recent version is C# 5.0, which was released on August 15, 2012.

C# is a general object-oriented programming (OOP) language for networking and Web development. C# is specified as a common language infrastructure (CLI) language.

In January 1999, Dutch software engineer Anders Hejlsberg formed a team to develop C# as a complement to Microsoft’s NET framework. Initially, C# was developed as C-Like Object Oriented Language (Cool). The actual name was changed to avert potential trademark issues. In January 2000, NET was released as C#. Its NET framework promotes multiple Web technologies.

C# improved and updated many C and C++ features, including the following:

* C# has a strict Boolean data variable type, such as bool, whereas C++ bool variable types may be returned as integers or pointers to avoid common programming errors.
* C# automatically manages inaccessible object memory using a garbage collector, which eliminates developer concerns and memory leaks.
* C# type is safer than C++ and has safe default conversions only (for example, integer widening), which are implemented during compile or runtime.

No implicit conversions between Booleans, enumeration members and integers (other than 0) may be converted to an enumerated type. User-defined conversions must be specified as explicit or implicit, versus the C++ default implicit conversion operators and copy constructors.

**SYNTAX:**

The core syntax of C# language is similar to that of other C-style languages such as C, C++ and Java. In particular:

* Semicolons are used to denote the end of a statement.
* Curly brackets are used to group statements. Statements are commonly grouped into methods (functions), methods into classes, and classes into namespaces.
* Variables are assigned using an equal’s sign but compared using two consecutive equal’s signs.
* Square brackets are used with arrays, both to declare them and to get a value at a given index in one of them.

1. **Why C#?**

C# is an elegant, simple, type-safe, object-oriented language that allows enterprise programmers to build a breath of applications.

C# also gives you the capability to build durable system-level components by virtue of the following features:

* Full COM/Platform support for existing code integration.
* Robustness through garbage collection and type safety.
* Security provided through intrinsic code trust mechanism.
* Full support of extensible metadata concepts.

**DISTINGUISHING FEATURES:**

Some notable features of C# that distinguish it from C and C++ (and Java, where noted) are:

* Portability
* Typing
* Meta programming
* Methods and functions
* Property
* Namespaces
* Memory access
* Exception
* Polymorphism

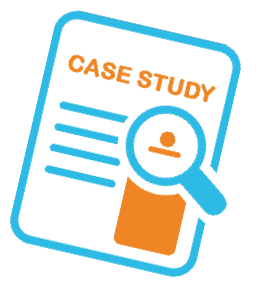




# REQUIREMENTS

**Requirement Specifications:**

* **Software requirements**
* Vuforia Support
* Android KitKat (4.4 or greater)
* **Hardware Requirements**
* Qualcomm Snapdragon 450 or above
* 2 GB ram
* Minimum disk space: 50 MB
* Rear Camera of a good quality



# **Case Study**

**Case Study:**

With all the Hardware and Software Requirements being known to us, we possessed all of the above given requirements and, downloaded and installed all the SDK’s and Softwares required.

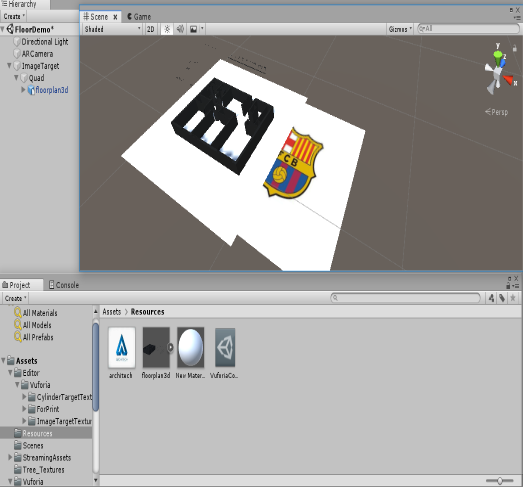
After acquiring all the assets, we developed a test application as a way of figuring out how can these technologies sync together and work cohesively for accomplishment of a single task.

In this case study, we first converted the 2D floorplan image of JPEG format into SVG file. Then we used Blender software to scale the height of the SVG image and made a 3D model of the image.

Then We took an image (“FC Barcelona logo”) and used it as a target image, upon which our 3D model created will be displayed in AR. We stored our target image in the Vuforia database.

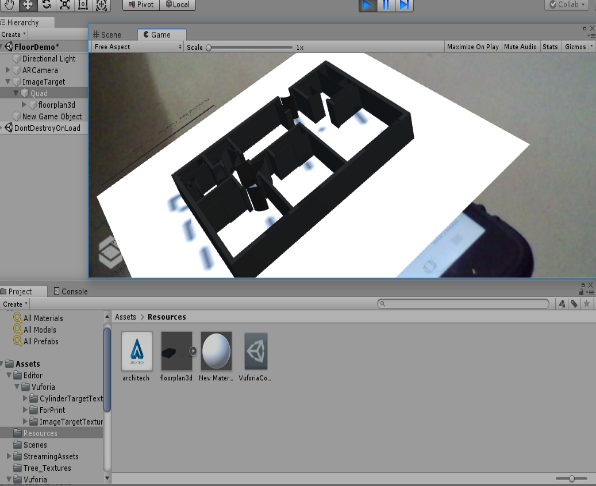
Vuforia generates a license key for every database created by the user. This license key is used by Unity, to retrieve the target images from specified database.

We made use of Vuforia plugin for Unity, which enables the use of a unique Component called AR Camera, which accesses the web cam of the PC and helps in Target image detection.



The results of this case study were:

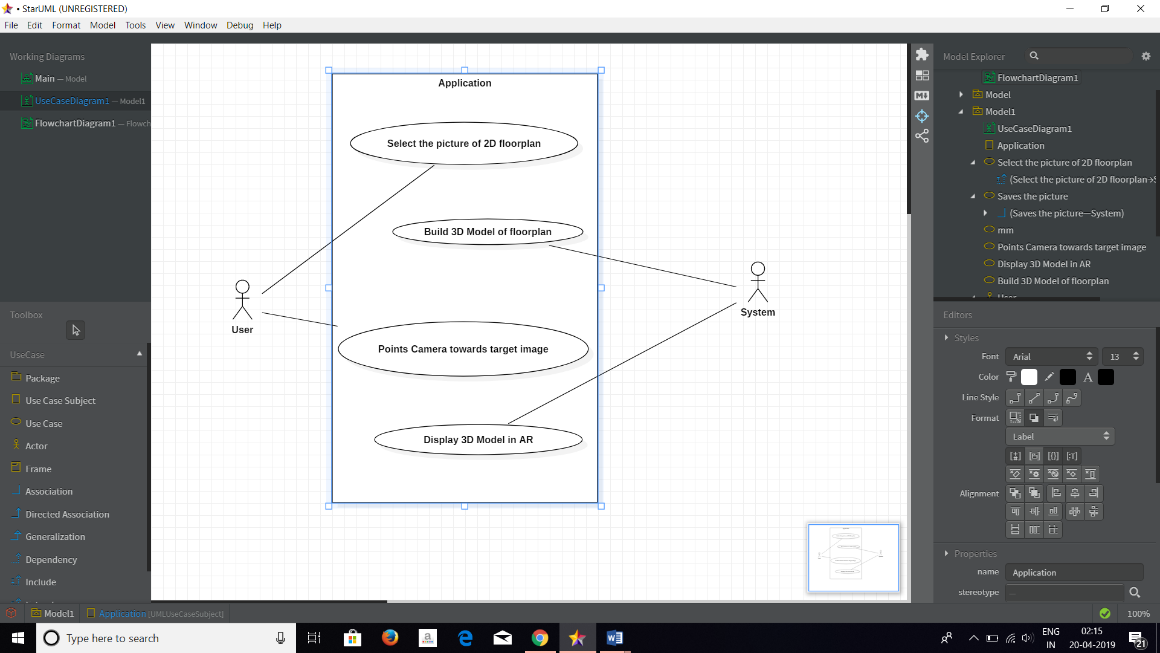
Upon detecting the Target image specified in the Vuforia database, the 3D model created with the help of blender got displayed atop of the Target image.

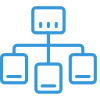




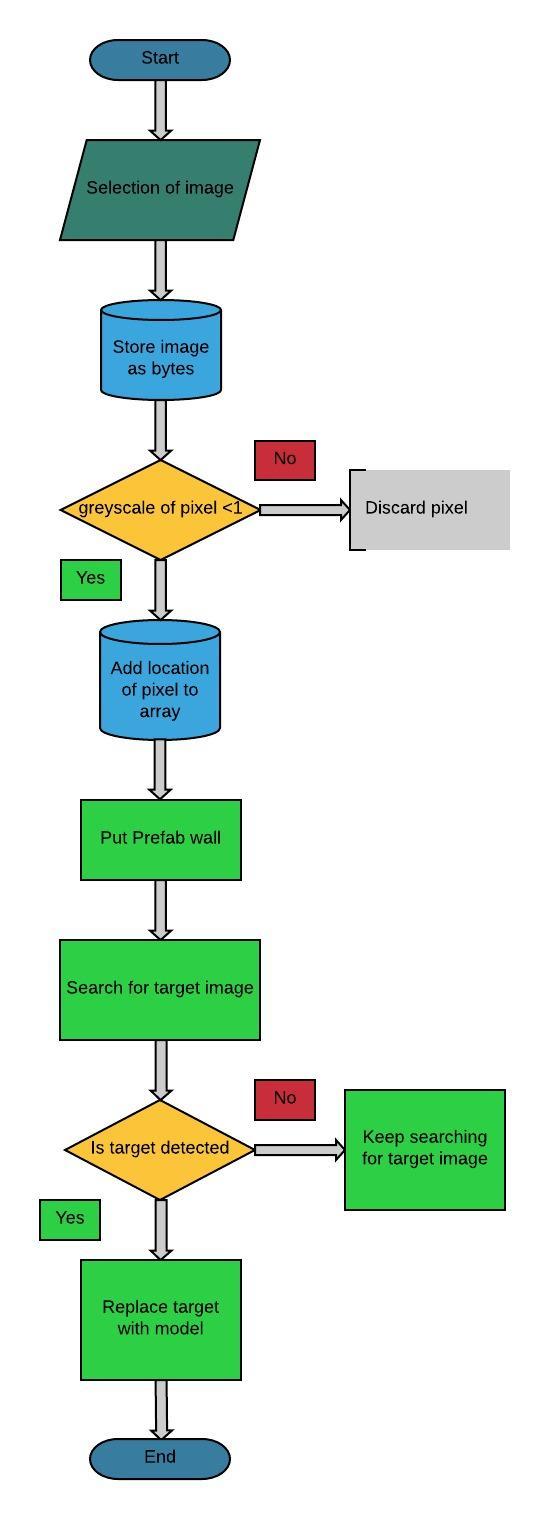
USE CASE

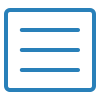
DIAGRAM





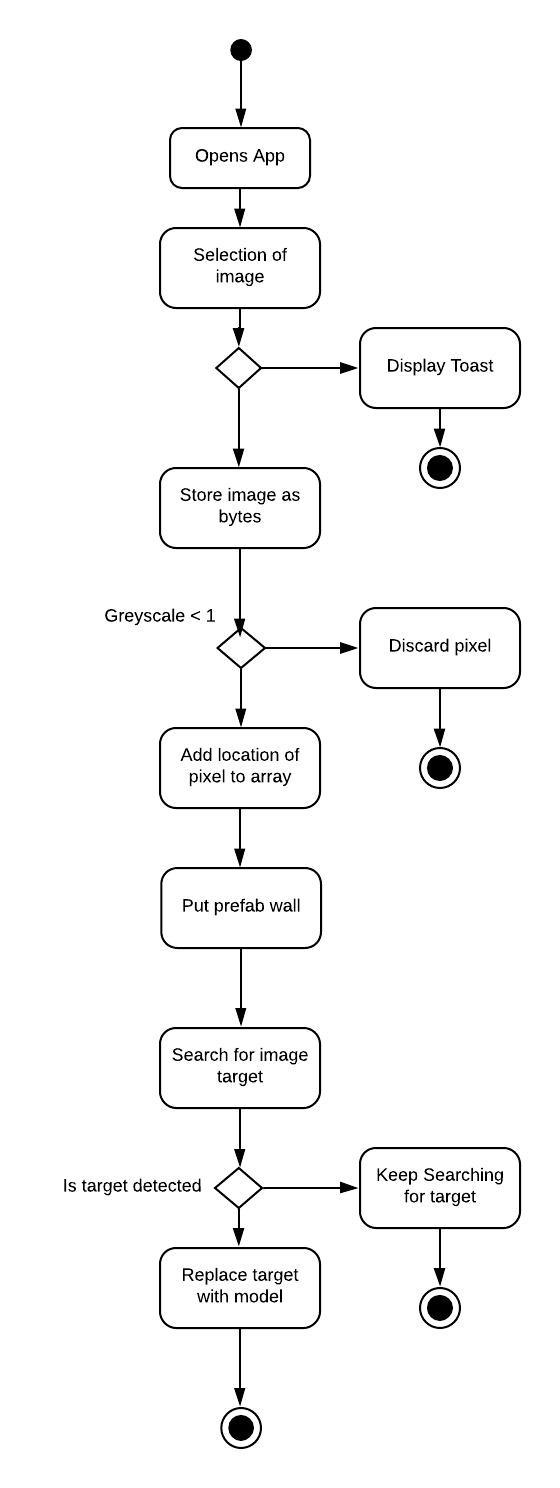
FLOWCHART

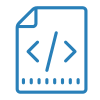




ACTIVITY

DIAGRAM





SOURCE CODE

**Manifest.xml:**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  [<manifest package="**com.example.androidnativelibrary**" xmlns:android="**http://schemas.android.com/apk/res/android**">](file:/C:/Users/Jash/Downloads/AndroidManifest.xml)<uses-permission android:name="**android.permission.READ\_EXTERNAL\_STORAGE**"/><uses-permission android:name="**android.permission.WRITE\_EXTERNAL\_STORAGE**"/><uses-permission android:name="**android.permission.CAMERA**"> </uses-permission><application> </application></manifest> |

**MyPrefabInstanciator.cs:**

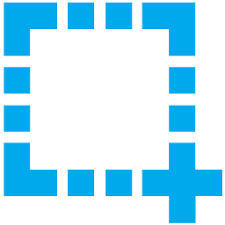
|  |
| --- |
| using UnityEngine;  using Vuforia;  using System.Collections;  using System;  using System.IO;  using System.Xml;  public class MyPrefabInstanciator : MonoBehaviour, ITrackableEventHandler  {  private TrackableBehaviour mTrackableBehaviour;  //public Transform myModelPrefab;  public GameObject prefabWall;  Texture2D img;  GameObject wall;  Vector3[] egpoints = new Vector3[600000];  int count = 0;  GameObject[] wallArray = new GameObject[600000];  // Use this for initialization  void Start()  {    mTrackableBehaviour = GetComponent<TrackableBehaviour>();  if (mTrackableBehaviour)  {  mTrackableBehaviour.RegisterTrackableEventHandler(this);  }  img = new Texture2D(1, 1);  byte[] imgData = File.ReadAllBytes(Application.persistentDataPath + "/floorplan.jpg");  img.LoadImage(imgData);  InitializeTexture();  }  public void InitializeTexture()  {  XmlTextWriter writer;  int i , j = 0;  Color c;  for (i = 0; i < img.width; i++)  {  for (j = 0; j < img.height; j++)  {  c = img.GetPixel(i, j);  if (c == Color.black)  {  egpoints[count] = new Vector3(i, j, 2f);  count++;  }  }  writer = new XmlTextWriter(Application.persistentDataPath + "/forloopInside.xml", null);  writer.WriteStartDocument();  writer.WriteStartElement("msg");  writer.WriteString("Width: " + img.width.ToString() + " Height: " + img.height + " Count: " + count.ToString() + "GreyScale: " + img.GetPixel(i,j).grayscale.ToString());  writer.WriteEndElement();  writer.WriteEndDocument();  writer.Close();  }  writer = new XmlTextWriter(Application.persistentDataPath + "/forlooputside.xml", null);  writer.WriteStartDocument();  writer.WriteStartElement("msg");  writer.WriteString("after load");  writer.WriteEndElement();  writer.WriteEndDocument();  writer.Close();    }  public void OnTrackableStateChanged(TrackableBehaviour.Status previousStatus, TrackableBehaviour.Status newStatus)  {  if (newStatus == TrackableBehaviour.Status.DETECTED || newStatus == TrackableBehaviour.Status.TRACKED)  {  OnTrackingFound();  }  else  {  OnTrackingLost();  }  }  private void OnTrackingFound()  {  if (true)  {  XmlTextWriter writer = new XmlTextWriter(Application.persistentDataPath + "/info.xml", null);  writer.WriteStartDocument();  writer.WriteStartElement("msg");  writer.WriteString(count.ToString());  writer.WriteEndElement();  writer.WriteEndDocument();  writer.Close();  for (int i = 0; i < egpoints.Length; i++)  {  //wall = (GameObject)Instantiate(prefabWall, new Vector3(mTrackableBehaviour.transform.position.x + points[i].x - 200, mTrackableBehaviour.transform.position.y - points[i].y - 400, mTrackableBehaviour.transform.position.z + points[i].z + 100) , Quaternion.identity);  wall = (GameObject)Instantiate(prefabWall, new Vector3(egpoints[i].x - 200, mTrackableBehaviour.transform.position.y - egpoints[i].y - 1100, mTrackableBehaviour.transform.position.z + egpoints[i].z + 250), Quaternion.identity);  wall.transform.parent = mTrackableBehaviour.transform;  //wall.transform.localPosition = new Vector3(0f, 0f, 0f);  wall.transform.localRotation = Quaternion.identity;  wall.transform.localScale = new Vector3(1f, 20f, 1f);  wall.SetActive(true);  //helps later to destroy walls when traking lost  wallArray[i] = wall;  }  }  }  void OnTrackingLost()  {  foreach (GameObject wall in wallArray)  {  Destroy(wall);  }  }  } |

**Demo.cs:**

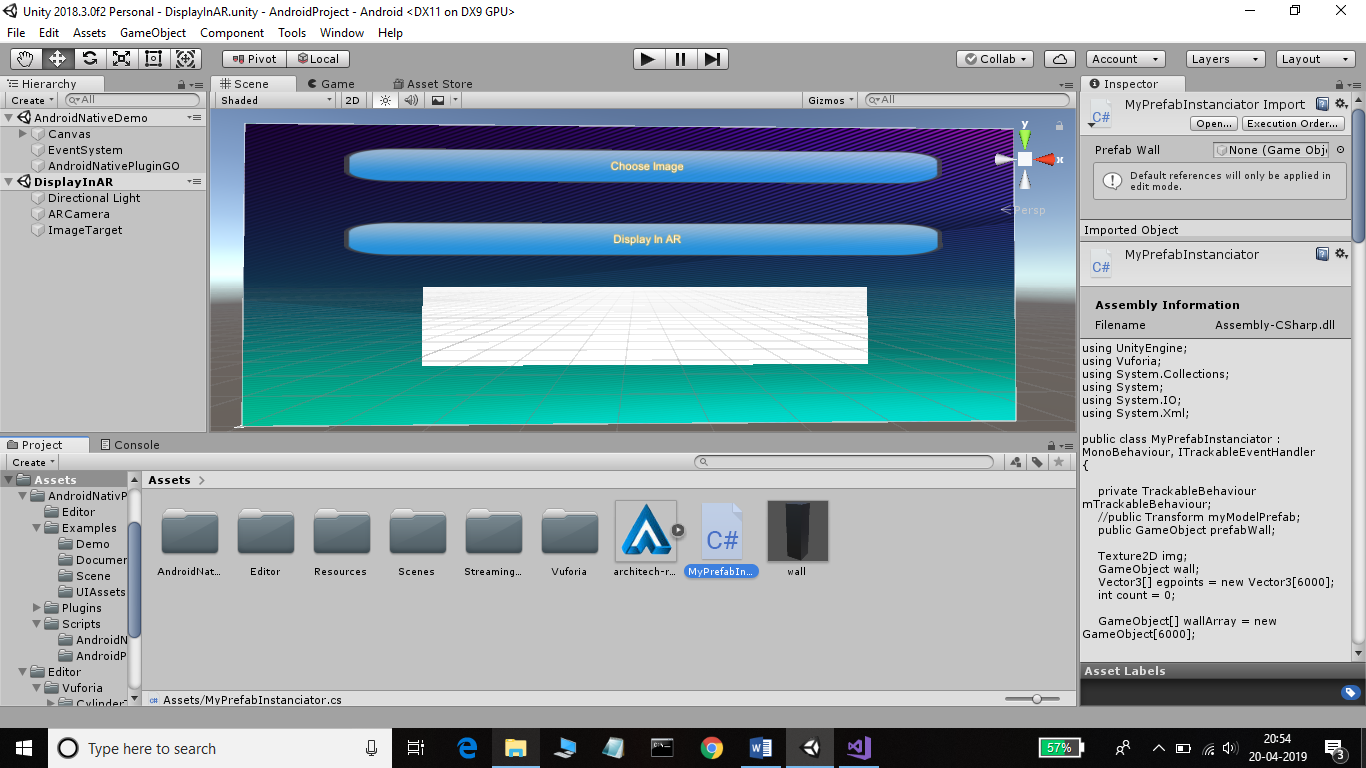
|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.UI;  using waqashaxhmi.AndroidNativePlugin;  using System.Xml;  using ConsoleApplication1;  using System.Drawing;  using System.IO;  namespace Default  {  public class Demo : MonoBehaviour {  public RawImage image;  // Use this for initialization  void Start() {  AndroidNativeController.OnFileSelectSuccessEvent = OnSuccess;  AndroidNativeController.OnFileSelectFailureEvent = OnFailure;  AndroidNativeController.OnPositiveButtonPressEvent = (message) => {  if (Application.platform == RuntimePlatform.OSXEditor || Application.platform == RuntimePlatform.WindowsEditor) {  #if UNITY\_EDITOR  UnityEditor.EditorApplication.isPlaying = false;  #endif  } else {  Application.Quit();  }  };  AndroidNativeController.OnNegativeButtonPressEvent = (message) => {  // Code whatever you want on click "NO" Button.  };  }  public void OnShowDialougeBoxButtonClick() {  AndroidNativePluginLibrary.Instance.ShowMessage("Level 1", "You have complete Level 1");  }  public void OnShowConfirmationButtonClick() {  // On Yes Button Click OnPositiveButtonPressEvent fire, and On "NO" button click OnNegativeButtonPressEventFire.  AndroidNativePluginLibrary.Instance.ShowConfirmationDialouge("Restart Game", "Do You Want to Restart the game.", "YES", "NO");  }  public void OnProgressBarButtonClick() {  AndroidNativePluginLibrary.Instance.ShowProgressBar("Loading Data", "Loading . . .", true);  }  public void OnShowToastButtonClick() {  AndroidNativePluginLibrary.Instance.ShowToast("Click On Toast Button");  }  public void OnOpenGallaryButtonClick() {  // after selecting file success OnSelectFile event fire  AndroidNativePluginLibrary.Instance.OpenGallary();  }  public void OnDislayInARClick() {  Program p = new Program();  p.NustaMethod();  }  private void OnSuccess(string path) {  AndroidNativePluginLibrary.Instance.ShowToast("File Selected:" + path, 1);  XmlTextWriter writer = new XmlTextWriter(Application.persistentDataPath + "/location.xml", null);  writer.WriteStartDocument();  writer.WriteStartElement("path");  writer.WriteString(path);  writer.WriteEndElement();  writer.WriteEndDocument();  writer.Close();  AndroidNativePluginLibrary.Instance.ShowToast(Application.persistentDataPath, 1);  StartCoroutine(ReadImage(path));  }  private void OnFailure(string err) {  AndroidNativePluginLibrary.Instance.ShowToast(err);  }  IEnumerator ReadImage(string path) {  WWW = new WWW("file://" + path);  yield return www;  image.texture = www.texture;  }  IEnumerator DismissProgressBar() {  yield return new WaitForSeconds(5f);  AndroidNativePluginLibrary.Instance.DismissProgressBar();  }  }  } |

**AndroidNativePluginLibrary.cs :**

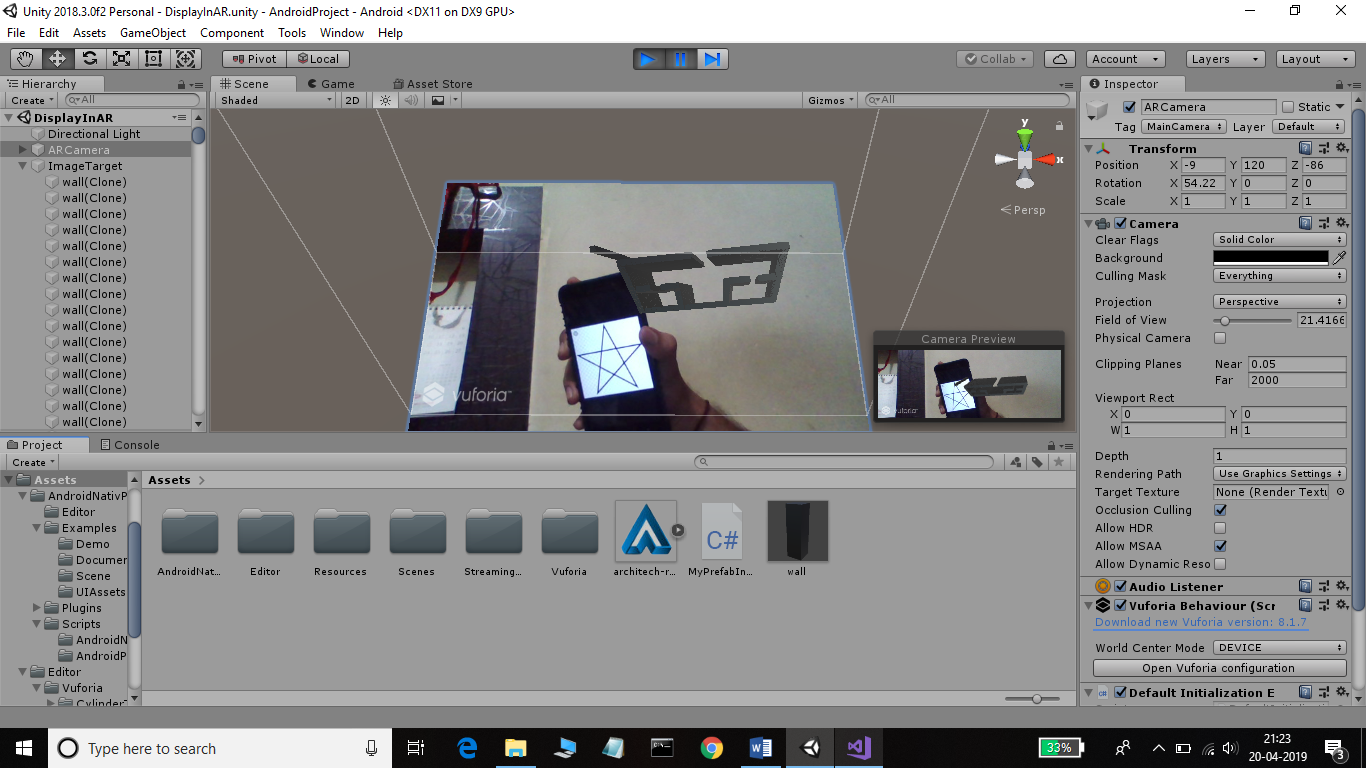
|  |
| --- |
| using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using System.Xml;  namespace waqashaxhmi.AndroidNativePlugin{  public class AndroidNativePluginLibrary : MonoBehaviour {  public static AndroidNativePluginLibrary Instance;  public string storage = Application.persistentDataPath;  #if UNITY\_ANDROID  private AndroidJavaClass androidObject;  #endif  // Use this for initialization  void Awake () {  Instance = this;  #if UNITY\_ANDROID  if (Application.platform == RuntimePlatform.Android) {  androidObject = new AndroidJavaClass ("com.example.androidnativelibrary.AndroidBridge");  }  #endif  }  #region ShowToast  /// <summary>  /// Shows the toast.  /// </summary>  /// <param name="message">Message.</param>  /// <param name="Length">Toast Duration, For Short 0,For Long 1.</param>  public void ShowToast(string message,int Length = 0){  #if UNITY\_ANDROID  androidObject.CallStatic("ShowToast", message,Length);  #endif  }  #endregion  #region OpenGallary  /// <summary>  /// Opens the gallary.  /// </summary>  public void OpenGallary(){  #if UNITY\_ANDROID  androidObject.CallStatic("OpenGallary");  #endif  }  #endregion  #region ShowMessage  public void ShowMessage(string title,string message,string buttonName= "OK"){  #if UNITY\_ANDROID  androidObject.CallStatic ("ShowMessage", title, message,buttonName);  #endif  }  #endregion  public void ShowConfirmationDialouge(string title,string message,string positiveButtonName= "Yes",string negativeButtonName= "NO"){  #if UNITY\_ANDROID  androidObject.CallStatic ("ConfirmationDialouge", title, message,positiveButtonName,negativeButtonName);  #endif  }  #region ShowProgressBar  /// <summary>  /// Shows the progress bar.  /// </summary>  /// <param name="title">Title.</param>  /// <param name="message">Message.</param>  /// <param name="cancelable">If set to <c>true</c> cancelable.On Click outside disable Progress</param>  /// <param name="style">Style.Horizontal = 1 Spinner = 2</param>  public void ShowProgressBar(string title ="",string message="",bool cancelable = true){  #if UNITY\_ANDROID  androidObject.CallStatic ("ShowProgressBar",title,message,cancelable);  #endif  }  public void DismissProgressBar(){  #if UNITY\_ANDROID  androidObject.CallStatic ("DismissProgressBar");  #endif  }  #endregion  }  } |



**Screenshots**



**Scene-1: GUI in Unity**



**Scene-2: Dynamic Model Displayed in AR in Unity**



ADVANTAGES

**Advantages:**

* Use in Real Estate:

This technology can bloom in Real Estate sector. Estate Agents can use this technology to increase their sales of flats/apartments. Customers can use it to get a better view of the property they’re interested in.

* Use in Interior Designing:

This technology can prove to be fruitful for Architects. They can use it to have a Three Dimensional view at the home they wish to design and plan their design for it accordingly.

* Ease in Decision Making:

In today’s world, real estate is a very costly investment. Buying, or even renting an apartment requires a large amount of capital. This technology will help the buyer in taking an appropriate decision about whether the apartment/flat is worth investing their money or not.

* Marketing:

Augmented Reality is a rising technology. Not many people having non-technical background are used to this technology. According to studies, the average age-group of the people intending to buy real estate is above 35. This generation of people, find technologies like Augmented Reality amusing. Thus, the estate agencies can use this technology as an marketing tool to attract more customers.

* Tour Guide:

It can be useful in Museums, or Amusement Parks to guide the visitors/tourists about the routes and attraction points via 3D Augmented model of the building itself. This can prove to be helpful while showing and explaining tourists the Emergency escape or exit routes.



LIMITATIONS

**Limitations:**

* The Floor-plan image which is to be converted into a 3D model has to be in Jpeg format. It does not scan images in any other file formats (such as PNG or TIFF).
* The Image to be selected should have High contrast pixels. If the pixels aren’t highly contrasted, the image scanning phase won’t function properly.
* The input image must have low resolution.



FUTURE SCOPE

**Future Scope:**

* Object Recognition:

The ability of Recognition of other objects; such as doors, windows and furniture, along with the floor plan and displaying them in Augmented Reality can be added.

* Virtual Interior Designing:

The users could Drag and Drop tables, chairs, sofas and other furniture in the AR model of the floor plan. They could design their home virtually and get the feel of how the house will look.

* Wall Decoration:

The user could choose which color they want the walls to be, select and apply the chosen color or pattern to the Augmented model. User can customize the wall according to their liking too and view it on Augmented Reality model.

* Saving And Printing 3D Models:

The user can generate a 3D model for not only floor plans but also, for other images he wishes to view in 3D. The user can later save the model as an .fbx file which can be used later for printing from a 3D printer.



CONCLUSION

In this project, we demonstrated some conclusive results on our 3-phase recognition approach to pseudo 3D building generation from 2D floor plan.

Our system based itself on the idea of using existing concepts, algorithm, libraries and technologies with minimal modifications and integrating them into a fully functional working application. Hence, we used C# and OpenCV for image processing and recognition, Unity engine for 3D model generation & Vuforia to display model in Augmented Reality.

The 3d model needs to be enhanced to produce more realistic model, enabling users to change color, textures, light and the ability to interactively insert objects like tables, chairs etc in the model generated

.We used existing image processing techniques like convert image into texture2d format & process each pixel, calculate greyscale of each pixel and use that greyscale value to instantiate wall.

The 3D Generation module provides a sense of aesthetic and a realistic scene in the 3D environment for the user’s experience with the system. It provides interactive and navigational capabilities with different views.



REFERENCES

https://unity3d.com/learning-c-sharp-in-unity-for-beginners

https://docs.unity3d.com/ScriptReference/

https://www.youtube.com/watch?v=BNh6WtozzeQ

https://answers.unity.com/questions/1063640/reference-dlls.html

https://developer.vuforia.com

https://assetstore.unity.com/packages/tools/integration/android-native-plugin-free-103738

https://youtu.be/ae6mW74FZSU

https://youtu.be/lFp8Z\_3wa7M

https://stackoverflow.com/questions/37157797/how-to-make-a-dynamic-model-in-unity-3d