BAB V

IMPLEMENTASI SISTEM

5.1 Instalasi dan Konfigurasi Sistem

Setelah melakukan analisa dan perancangan terhadap Aplikasi Media Pembelajaran Alat Musik Daerah di Indonesia menggunakan Teknologi *Augmented Reality,* maka tahapan selanjutnya adalah implementasi terhadap aplikasi yang telah dibuat. Pada tahapan ini ada beberapa bahasan yaitu :

a. Spesifikasi kebutuhan sistem meliputi perangkat keras (*Hardware*) dan perangkat lunak (*Software*).
b. Instalasi perangkat lunak
c. Konfigurasi system operasi dan perangkat lunak
d. Implementasi *Augmented Reality* dengan *Smartphone* yang meliputi perancangan antarmuka.

5.1.1 Spesifikasi Sistem

Spesifikasi system terdiri dari spesifikasi perangkat keras *(Hardware)* dan perangkat lunak *(Software)* yang digunakan untuk memenuhi kebutuhan mengimplementasikan Aplikasi Media Pembelajaran Alat Musik Daerah di Indonesia menggunakan Teknologi *Augmented Reality.* Berikut ini adalah spesifikasi dari kebutuhan system yang digunakan :

a. Perangkat Keras *(Hardware)*

1) Processor *Intel Core* i3
2) RAM 4 GB
3) Harddisk 500 GB
4) Smartphone dengan minimum versi android 5.1 (lollipop)

b. Perangkat Lunak *(Software)*

1) Sistem Operasi : *Windows 10 Home*
2) Objek 3D : *Blender*

3) *Tools Augmented Reality* : *Unity* 2019.1.2f1, *Vuforia Augmented Reality Support*

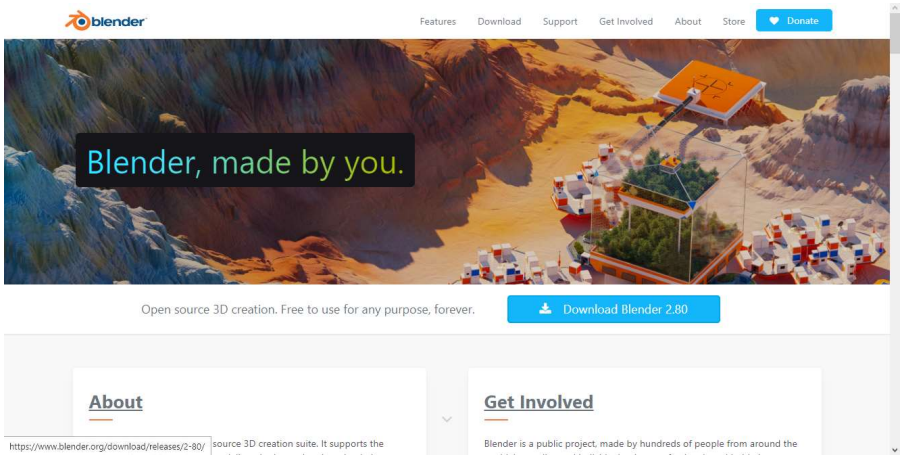
4) *Android Tools* : *Android* SDK & NDK *Tools*, *Open* JDK

5.2.2 Instalasi dan Konfigurasi Software

Sistem aplikasi yang akan berjalan membutuhkan perangkat lunak *(Software)*. Perangkat lunak tersebut harus menjalani proses instalasi dan konfigurasi sebelum digunakan untuk mengimplementasikan Aplikasi Media Pembelajaran Alat Musik Daerah di Indonesia menggunakan Teknologi *Augmented Reality.* Berikut dibawah ini dijelaskan cara instalasi dan konfigurasinya.

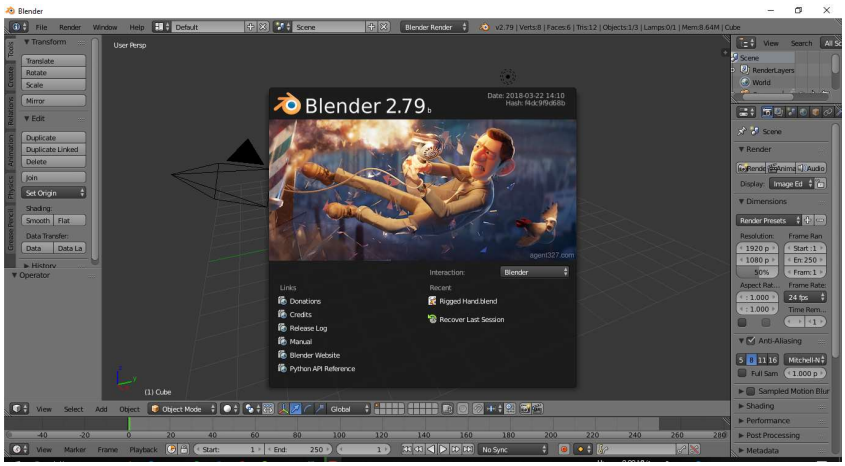
a. Instalasi *Blender*

Aplikasi *Blender* yang digunakan adalah *Blender* 3D. Aplikasi ini dapat di*download* melalui link https://www.blender.org. berikut tampilan *website* tersebut.



Gambar 5.1 Tampilan *website* dari Blender

Klik menu download *blender* 2.80 pada halaman website diatas untuk mendapatkan aplikasi *blender* 2.80. Setelah itu proses download akan berjalan. Langkah selanjutnya menginstall aplikasi *blender* 2.80 pada laptop yang akan digunakan untuk membuat objek 3D. Spesifikasi yang dibutuhkan untuk menginstall aplikasi ini minimal menggunakan RAM sebesar 4 Gb dengan sistem operasi minimal *windows* 7. Gambar tampilan awal saat Blender di jalankan ada pada halaman berikutnya.

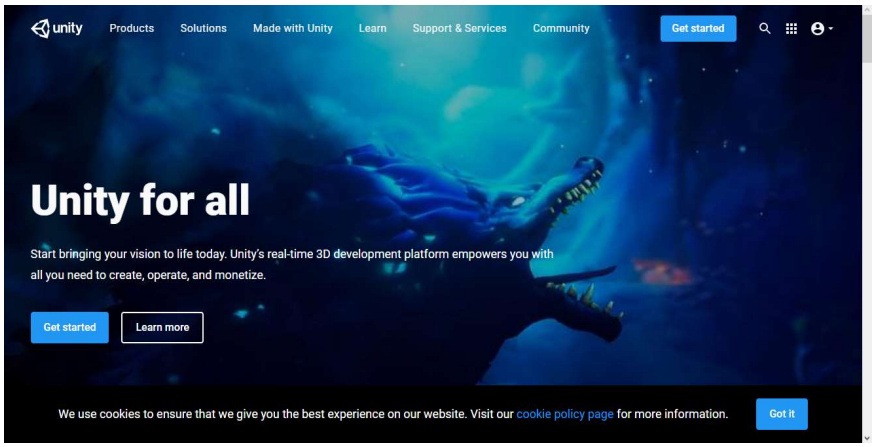


Gambar 5.2 Halaman awal aplikasi *Blender*

Pada gambar 5.2 adalah tampilan awal saat membuka aplikasi *blender*. Untuk memulai projek dapat dilakukan dengan cara mengklik pada halaman kerja.

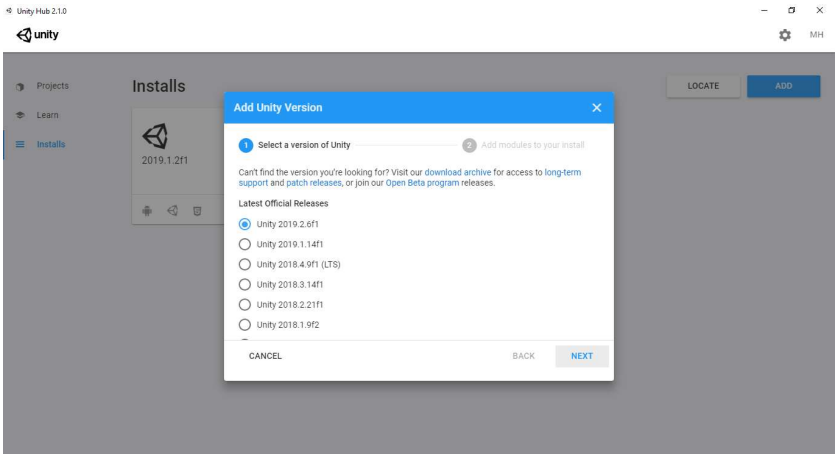
b. Instalasi dan Konfigurasi *Unity* 2019.1.2f1.

Aplikasi yang digunakan adalah *Unity* 2019.1.2f1 versi 64 bit. Untuk dapat mendownloadnya, terlebih dahulu yang dilakukan adalah mengakses websitenya melalui link https://unity.com.



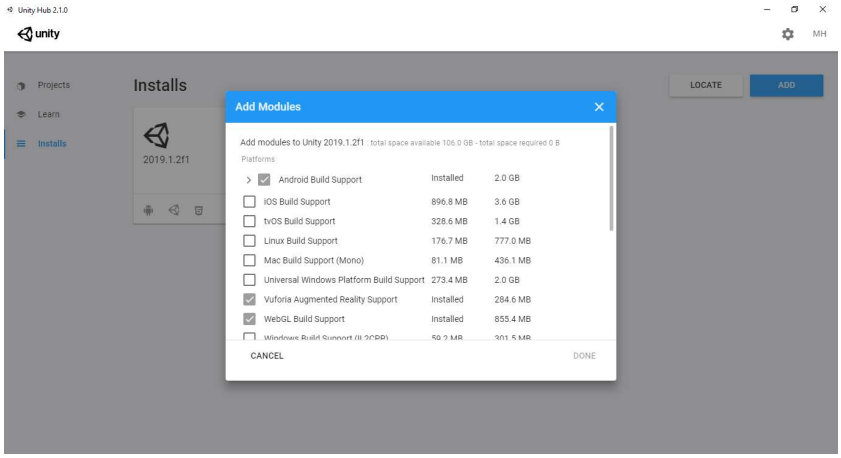
Gambar 5.3 tampilan website *Unity*

Setelah muncul halaman utama dari unity, maka selanjutnya download *Unity Hub*. *Unity Hub* berfungsi untuk mengkonfigurasi Component apa saja yang akan digunakan untuk membuat sebuah projek atau sistem. Berikut tampilan dari *Unity Hub*.



Gambar 5.4 Halaman awal dari *Unity Hub*

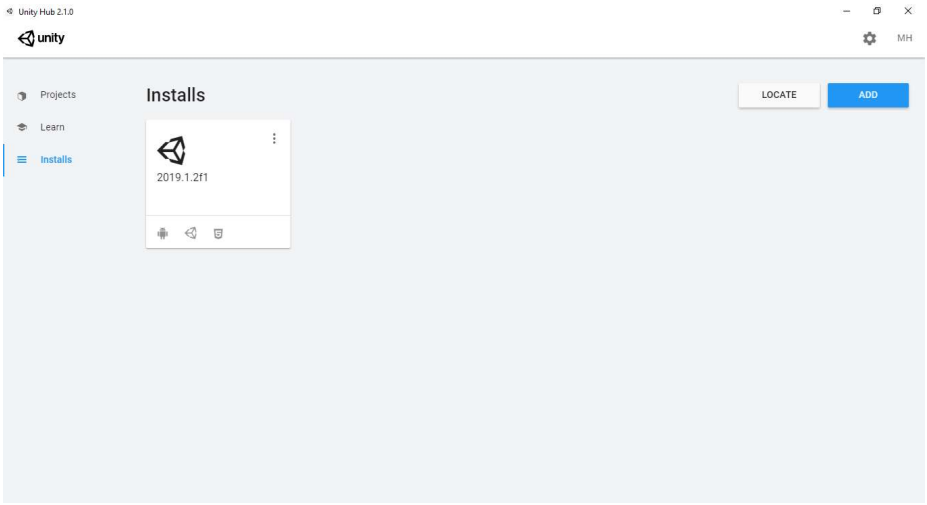
Selanjutnya dilakukan konfigurasi untuk component yang dibutuhkan. Berikut tampilan halaman konfigurasi component dalam *Unity*. Untuk versi yang digunakan menggunakan versi 2019.2.f1 dikarenakan sudah mengalami berbagai pembaharuan dari berbagai sector seperti tampilan aplikasi, letak tools dan pembaharuan component – component yang suda dapat didownload secara langsung.



Gambar 5.5 Halaman konfigurasi *Component* dalam *Unity*

Component yang digunakan antara lain berupa Android Build Supoort yang berisi SDK dan NDK *Tools*, *Vuforia Augmented Reality*, dan *WebGL Build Support*. Component tersebut didapatkan dengan cara mendownloadnya melalui aplikasi penghubung *Unity* yaitu *Unity Hub* yang berfungsi untuk mengkonfigurasi

component yang akan digunakan dalam *Unity*. Untuk mendapatkan component – component tersebut, aplikasi unity hub harus terhubung ke jaringan internet.



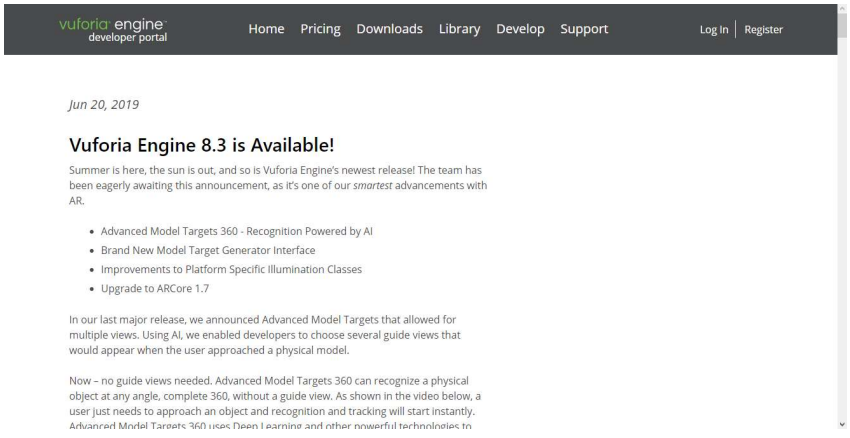
Gambar 5.6 Tampilan Unity 2019.1.2f1 sudah terinstall

Pada gambar 5.6 adalah tampilan Unity yang sudah terdownload dan terinstall dengan component – component yang dibutuhkan. Disini dapat kita tambahkan beberapa unity dengan versi berbeda dengan component – component yang berbeda – beda juga.

5.2 Implementasi

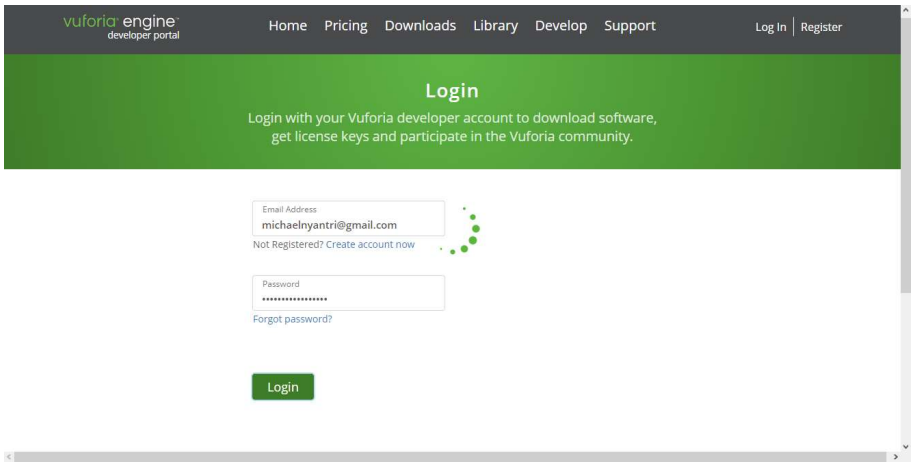
5.2.1 Implementasi *Vuforia*

Vuforia digunakan untuk menyimpan gambar yang nantinya akan di*import* sebagai marker**.** Berikut adalah tampilan halaman website dari *Vuforia Developer*.



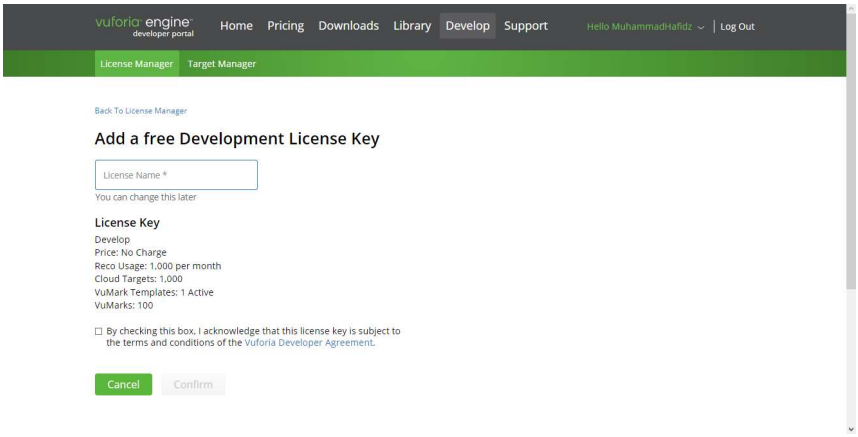
Gambar 5.7 Halaman Vuforia Developer

Pada gambar 5.7 adalah tampilan halaman website *Vuforia Developer*. Selanjutnya yang dilakukan adalah register terlebih dahulu. Kemudian lakukan *Register* dan *Login* setelah mendapatkan konfirmasi dari *email* yang didaftarkan. Setelah itu akan dialihkan ke halaman berikut seperti gambar 5.8



Gambar 5.8 Halaman Login

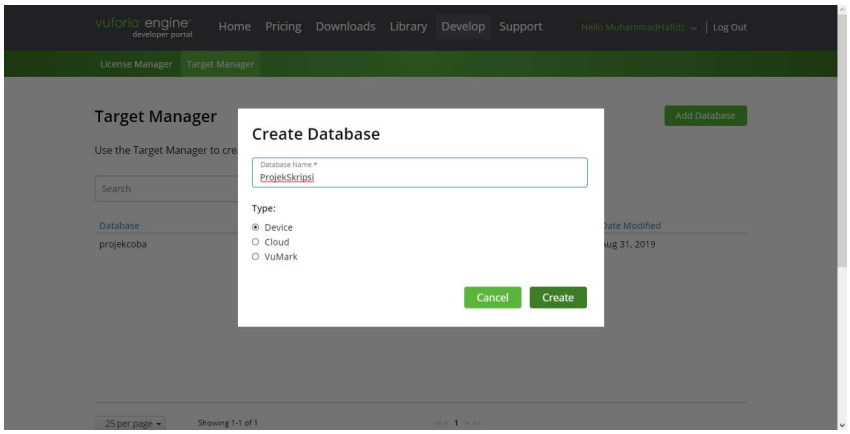
Untuk dapat mengupload gambar yang akan dijadikan *marker*, terlebih dahulu membuat License Manager untuk projek *Augmented Reality* yang akan kita buat.



Gambar 5.9 Halaman *License Manager*

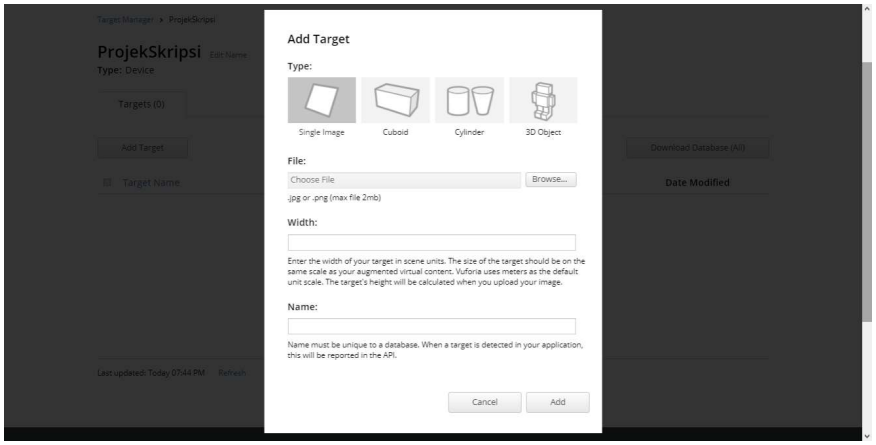
Setelah mendapatkan license manager, selanjutnya adalah mengupload gambar yang akan digunakan sebagai marker ke website vuforia developer. License key ini dapat digunakan secara free atau gratis dengan pemakaian yang sudah dibatasi sebanyak 1000 kali per bulan. Pada kolom license name dapat diisi sesuka hati si

pengguna atau dapat diisi dengan nama yang sama dengan projek yang sedang atau akan dikerjakan.



Gambar 5.10 Halaman *Create Database*

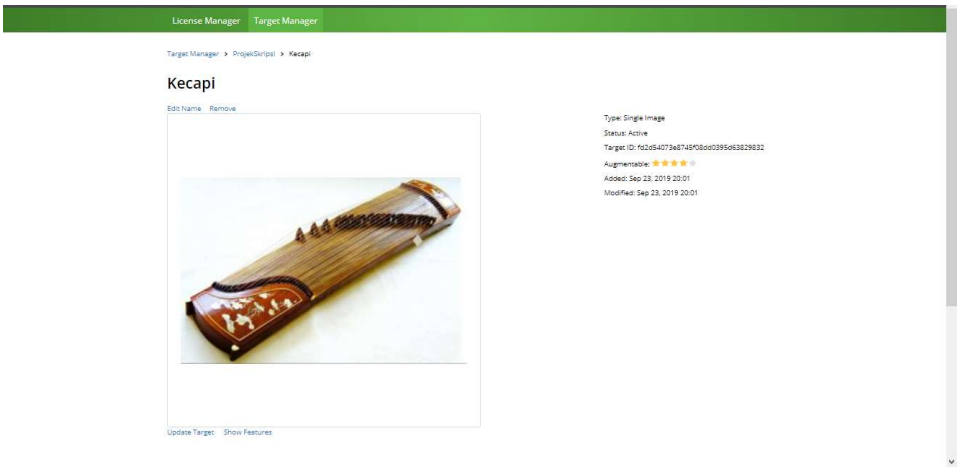
Dalam membuat *database* pada target manager dimulai dengan *Add Database* dan memberi nama kemudian memilih *Type Database.* Karena disini media yang akan digunakan adalah Smarthphone, maka dipilih type database yang Device. Setelah databasenya dibuat, maka dapat langsung menggunakannya untuk mengupload gambar yang akan digunakan sebagai marker.



Gambar 5.11 Halaman *Add Target*

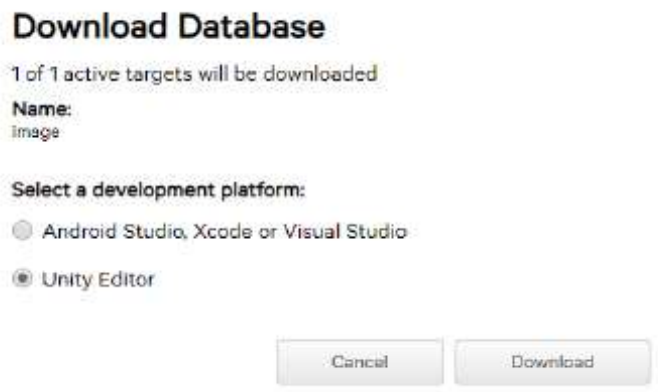
Setelah berhasil klik nama database yang anda buat klik add target lalu isi form Type pilih Single Image, tentukan lokasi file berada, isi Width menjadi 6 dan beri nama sesuai yang anda inginkan lalu klik add. Setelah berhasil anda lihat berapa rating gambar bila kurang dari 4 bintang ganti gambar yang lain karena

kemungkinan besar model 3d tidak akan muncul di target gambar anda. Berikut contohnya



Gambar 5.12 Image yang sudah terupload

Pada gambar 5.11 adalah contoh image untuk dijadikan sebagai marker yang sudah terupload ke vuforia developer. Image tersebut sudah memenuhi syarat untuk menjadi marker.

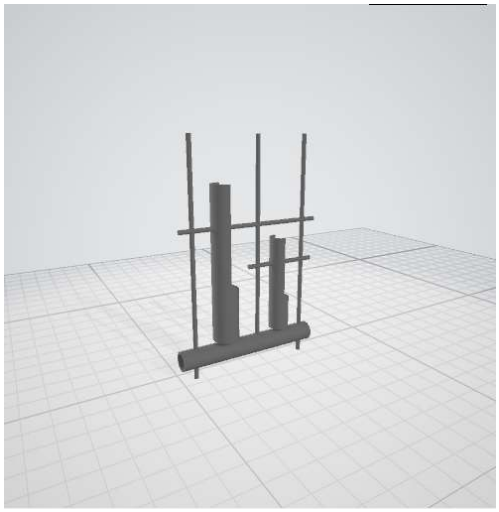


Gambar 5.11 Halaman download database

Selanjutnya klik download *database*. Pilih *Unity editor.* dikarenakan kita akan menggunakan Unity3D, maka pilih *Unity Editor*. setelah berhasil terdownload, maka langkah selanjutnya yaitu membuat objek 3D yang akan digunakan untuk melengkapi pembuatan projek aplikasi dengan teknologi *augmented reality*. Objek 3D yang dapat digunakan pada *unity* menggunakan format file *.obj*

5.2.2 Implementasi Pembuatan Objek dengan Blender

Implementasi pembuatan objek 3D ini menggunakan *software blender* 27b. berikut adalaha gambar yang menghasilkan implementasi pembuatan objek 3D pada *blender.*



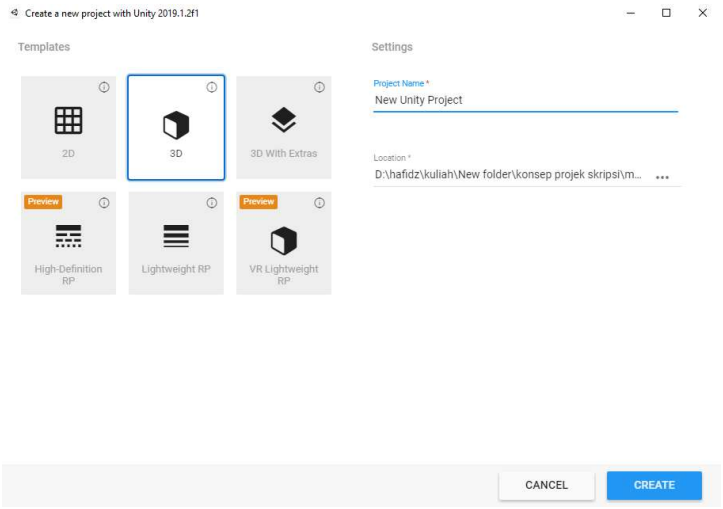
Gambar 5.13 Hasil Objek 3D dari blender 3D

Pada gambar 5.12 adalah hasil pembuatan onjek 3D dengan aplikasi *Blender 3D*. objek 3D yang dihasilkan berupa objek dengan tekstur yang sederhana dan berformat file .obj. Selanjutnya objek 3D tersebut disave dan siap untuk melengkapi pembuatan projek aplikasi dengan *Augmented Reality*.

5.2.3 Implementasi Unity

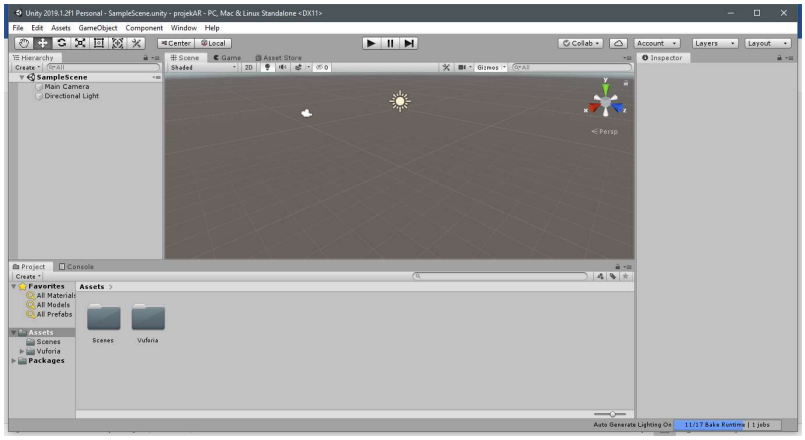
Implementasi pembuatan *Augmented Reality* ini menggunakan *Unity* 2019.1.2f1(64–bit). Pertama yang harus dilakukan adalah *create project* pada *software* unity. Karena project yang akan dikerjakan berhubungan dengan 3D, maka untuk templatenya dipilih yang 3D. Setelah selesai memilih *template* yang akan digunakan, langkah selanjutnya klik *create* seperti pada gambar 5.13. Tunggu sampai proses loading pemuatan aplikasi Unity sampai selesai. Proses ini berisi proses importing *modules*, *script* dan *scene – scene* yang akan digunakan ketika mengerjakan sebuah projek. Ketika sudah selesai, maka akan muncul tampilan halaman awal lembar kerja projek seperti pada gambar 5.14 yang ada pada halaman berikutnya.

Berikut tampilan dari halaman awal *create new project* pada *unity 2019.1.2f1*. nama untuk *project* dapat disesuaikan dengan keinginan dari *user* atau projek yang akan dikerjakan.



Gambar 5.14 *Create Project*

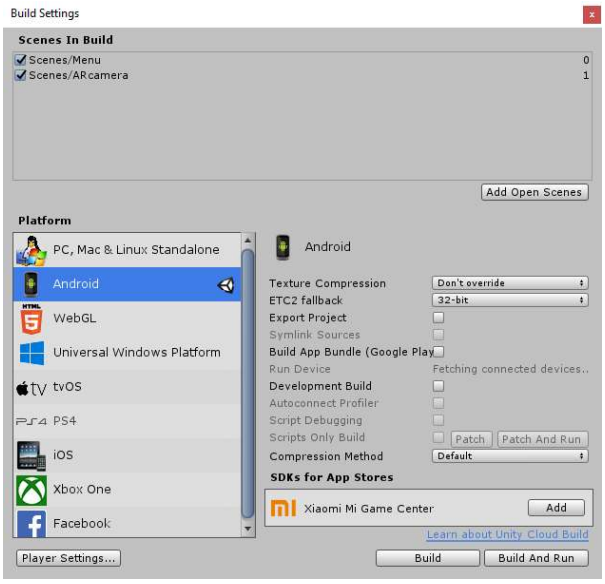
Klik create untuk mulai membuat projek tersebut. Pada gambar 5.14 adalah halaman awal untuk lembar kerja projek yang sudah kita buat sebelumnya.



Gambar 5.15 Tampilan Projek Awal dari *Unity*

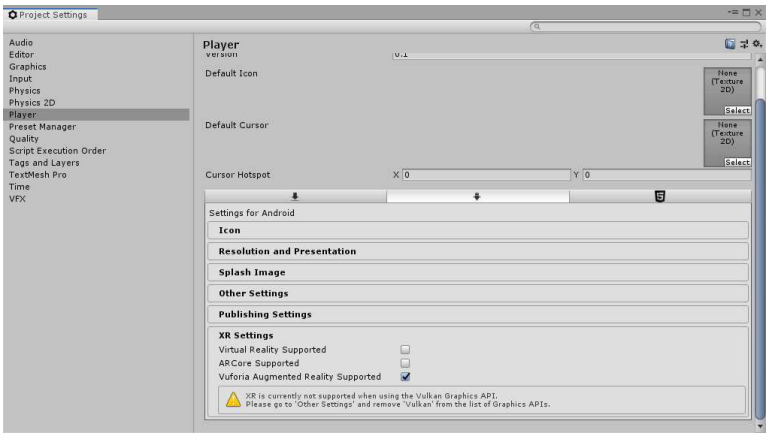
Setelah membuat project baru, selanjutnya melakukan setting pada Build Settings ke Android seperti gambar berikut. Sebelum import database ke Unity3D kita konfigurasi dulu. Klik File > Build Settings. Klik Android > Switch Platform

karena kita akan menggunakan android. Tunggu beberapa saat hingga proses selesai.



Gambar 5.16 *Build Setting Android*

Tetap di Build Setting untuk mengaktifkan Library Augmented Reality dari Vuforia klik Player Setting. Tampilan paling kanan berubah cari XR Setting centang Vuforia Augmented Reality.

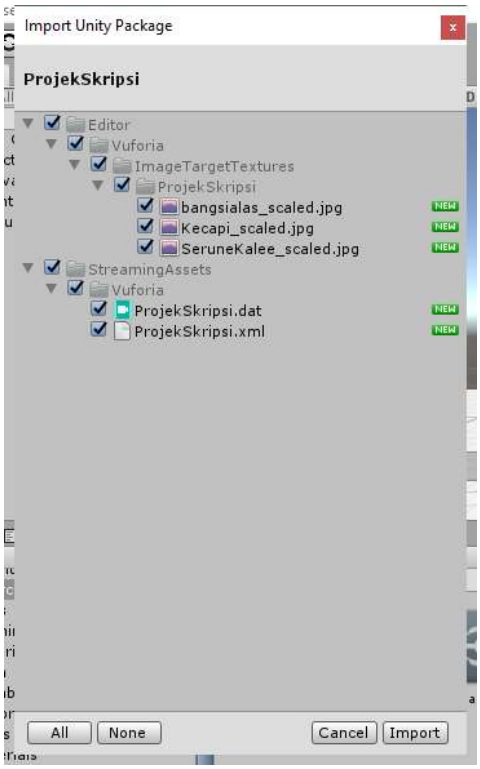


Gambar 5.17 *Player Setting*

Sekarang kita masukkan license dan databasenya. Klik ARCamera di pojok kiri window Hierarchy kemudian ke pojok kanan di window Inspector klik “Open Vuforia Configuration”. Pertama Masukkan *License* yang telah dibuat tadi. Caranya masuk ke https://developer.vuforia.com/vui/develop/licenses klik nama *license* lalu

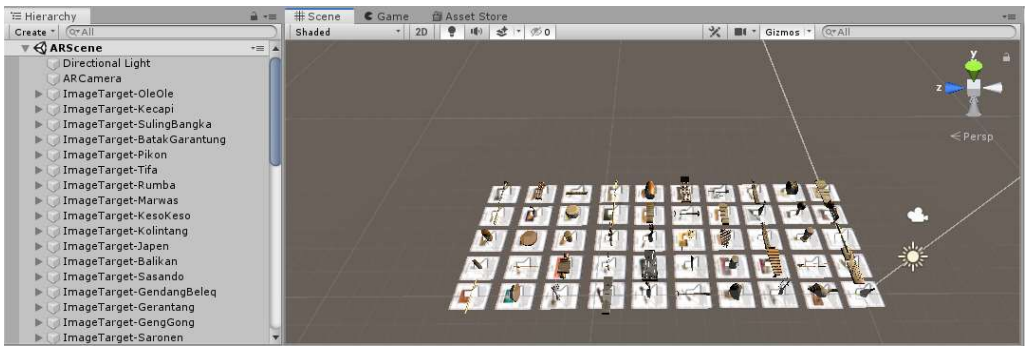
copy paste ke unity di kolom App License Key. Masukkan databasenya dengan cara klik 2x file yang telah di download dari web vuforia lalu klik import.

Gambar 5.18 *Add License Key*



Gambar 5.19 *Import Unity Package*

Jika sudah benar maka tekan *Import* dan akan berjalan proses Preparing Import Package. Jika sudah selesai proses importnya, maka akan muncul seperti gambar 5.20 pada halaman berikutnya.



Gambar 5.20 hasil *import package*

Gambar diatas merupakan tampilan dari ketika *unity* sudah melakukan proses import package. Langkah selanjutnya adalah mulai membuat tampilan dari aplikasinya.

5.3 Implementasi Antarmuka *Augmented Reality*

5.3.1 Halaman Splash Screen

Layar spash adalah elemen control grafis yang terdiri dari jendela yang berisi gambar, logo instansi kampus dari si pembuat aplikasi . Layar spash biasanya muncul saat sebuah program atau aplikasi diluncurkan atau dijalankan. Berikut layar spash yang muncul saat aplikasi dijalankan.

Gambar 5.21 tampilan halaman *Splash Screen*

Pada gambar 5.19 ditampilkan halaman splash screen berupa logo Universitas Bhayangkara Surabaya. Halaman *splash screen* ini muncul sebelum halaman menu utama dari aplikasi muncul. Halaman ini apat digunakan sebagai penunjuk identitas dari pembuat aplikasi.

5.3.2 Halaman Utama

Halaman utama dalam sistem ini adalah suatu halaman yang berguna untuk mengakses keseluruhan menu dalam sistem ini. Halaman utama akan muncul setelah pengguna melewati halaman *splash screen*. Berikut ini desain halaman utama dari sistem ini.

Gambar 5.22 Halaman Utama

Pada halaman utama terdapat beberapa tombol yang digunakan untuk berpindah-pindah antar halaman. Berikut ini penjelasan dari tombol-tombol tersebut.

a. Tombol *Help*

   Berfungsi untuk memberitahu petunjuk cara penggunaan aplikasi.

b. Tombol *Start Scan*

   Berfungsi untuk menscan marker yang disediakan untuk memunculkan objek 3D dari marker tersebut.

c. Tombol Soal – Soal

   Berfungsi untuk melatih kemampuan si pengguna dalam mengingat dan mempelajari alat music daerah dari aplikasi tersebut.

d. Tombol *About*

   Berfungsi menampilkan isi tentang aplikasi dan data diri singkat pembuat aplikasi

e. Tombol *Exit*

   Berfungsi untuk keluar dari aplikasi

Dan berikut Source Code dari Halaman Utama

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class UIController : MonoBehaviour
{
    public bool mute;
    public float scalingSpeed = 0.03f;
    public float rotationSpeed = 70.0f;
    public GameObject Information;
    public GameObject[] Tools;
    public bool rotDown, rotUp, rotRight, rotLeft, scaleUp,
scaleDown;
    public Text infoText;
    public AudioSource bunyiAudio;
    public AudioSource infoAudio;
    public AudioSource buttonAudio;
    public AudioClip tap, close;
    private void Update()
    {
        if (rotDown!=false)
        {
            for (int i = 0; i < Tools.Length; i++)
            {
                Tools[i].transform.Rotate(rotationSpeed  *
Time.deltaTime, 0, 0);
            }
        }
        if (rotUp!=false)
        {
            for (int i = 0; i < Tools.Length; i++)
            {
                Tools[i].transform.Rotate(-rotationSpeed *
Time.deltaTime, 0, 0);
            }
        }
        if (rotRight!=false)
        {
            for (int i = 0; i < Tools.Length; i++)
            {
                Tools[i].transform.Rotate(0,             -
rotationSpeed * Time.deltaTime, 0);
            }
        }
        if (rotLeft!=false)
        {
            for (int i = 0; i < Tools.Length; i++)
            {
                Tools[i].transform.Rotate(0,  rotationSpeed
* Time.deltaTime, 0);
```

```
            }
        }
        if (scaleUp!=false)
        {
            for (int i = 0; i < Tools.Length; i++)
            {
                Tools[i].transform.localScale    +=    new
Vector3(scalingSpeed, scalingSpeed, scalingSpeed);
            }
        }
        if (scaleDown!=false)
        {
            for (int i = 0; i < Tools.Length; i++)
            {
                Tools[i].transform.localScale    -=    new
Vector3(scalingSpeed, scalingSpeed, scalingSpeed);
            }
        }
    }
    public void RotationDown(bool aktif)
    {
        audioClick();
        rotDown = aktif;
    }
    public void RotationUp(bool aktif)
    {
        audioClick();
        rotUp = aktif;
    }
    public void RotationRight(bool aktif)
    {
        audioClick();
        rotRight = aktif;
    }
    public void RotationLeft(bool aktif)
    {
        audioClick();
        rotLeft = aktif;
    }
    public void ScaleUp(bool aktif)
    {
        audioClick();
        scaleUp = aktif;
    }
    public void ScaleDown(bool aktif)
    {
        audioClick();
        scaleDown = aktif;
    }
    public void Refresh()
    {
        audioClick();
```

```
        for (int i = 0; i < Tools.Length; i++)
        {
            Tools[i].transform.localScale = new Vector3(1,
1, 1);
            Tools[i].transform.localEulerAngles    =    new
Vector3(0, 0, 0);
        }
    }
    public void changeScene(string sceneName)
    {
        audioClick();
        SceneManager.LoadScene(sceneName);
    }
    public void showInfo()
    {
        if (!infoAudio.isPlaying)
        {
            audioClick();
            Information.SetActive(true);
            infoAudio.Play();
        }
    }
    public void muteAudio()
    {
        audioClick();
        if (mute != true)
        {
            bunyiAudio.Pause();
            mute = true;
        }
        else {
            bunyiAudio.Play();
            mute = false;
        }
    }
    public void closeInfo()
    {
        audioClose();
        infoAudio.Stop();
    }
    void audioClick()
    {
        buttonAudio.clip = tap;
        buttonAudio.Play();
    }
    void audioClose()
    {
        buttonAudio.clip = close;
        buttonAudio.Play();
    }
}
```

Berikut penjelasan mengenai source code diatas:

a) `Halo.Play();`
   // memainkan audio opening
   `yield return new WaitForSeconds(0.5f);`
   // audio diputar dengan durasi 0.5 detik
   `SelamatDatang.Play();`
   // memainkan audio dengan nama selamat datang

5.3.3 Halaman Help

Tampilan halaman *help* akan tampil ketika menu *Help* pada halaman beranda ditekan. Halaman *Help* berisi pemaparan tentang tata cara atau panduan penggunaan dari aplikasi ini. Implementasi tampilan halaman beranda aplikasi dapat dilihat pada Gambar 5.23 berikut ini.

Gambar 5.23 Halaman *Help*

Berikut source code dari halaman *Help*

```
public void TampilBantuan()
    {
        if (GameObject.FindWithTag("Bantuan") != null)
        {
            if (ban)
            {
GameObject.FindWithTag("Bantuan").GetComponent<Canvas>().e
nabled = true;
GameObject.FindWithTag("Bantuan").transform.GetChild(0).ga
meObject.active                    =                    false;
GameObject.FindWithTag("Bantuan").transform.GetChild(0).ga
meObject.active = true;
                ban = false;
            }
            else
```

```
            {
                var                 animator                =
GameObject.FindWithTag("Bantuan").transform.GetChild(0).Ge
tComponent<Animator>();
                if
(animator.GetCurrentAnimatorStateInfo(0).IsName("Open"))
                    animator.Play("Close");
StartCoroutine(RunPopupDestroy(GameObject.FindWithTag("Ban
tuan").transform.GetChild(0).gameObject));
                ban = true;
            }
        }
    }
```

Berikut penjelasan mengenai source code diatas :

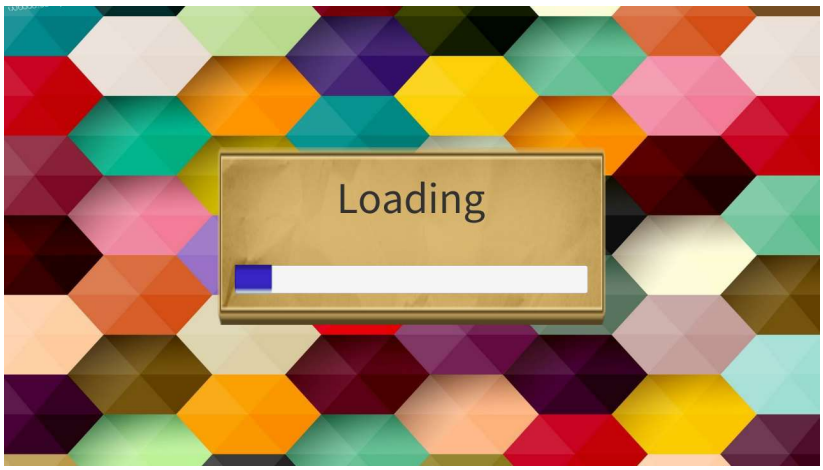
a) **`GameObject.FindWithTag("Bantuan")`** / /class dasar dengan nama bantuan

b) **`GameObject.FindWithTag("Bantuan").GetComponent<Canvas>().enabled = true;`** //menampilkan komponen objek dengan label nama bantuan

c) **`If(animator.GetCurrentAnimatorStateInfo(0).IsName("Open"))`** // mendapatkan daftar *clip* info saat keadaan *open*

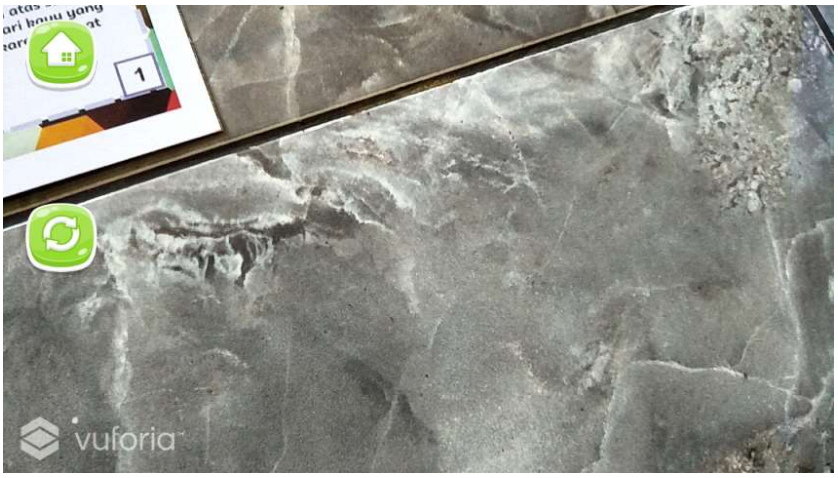
5.3.4 Halaman Start Scan

Halaman *start scan* merupakan tampilan kamera pada *smartphone* android. Pada layar ini *user* mencari penanda (*marker*), kamera akan terus melakukan pelacakan *marker* sampai ditemukannya penanda yang sesuai. Ketika penanda terdeteksi dan berhasil dibaca oleh sistem maka pada layar inilah *user* dapat melihat objek 3 dimensi yang ditampilkan tepat di atas *marker* yang terdeteksi. Untuk halaman *Start Scan* pada buku *marker*, saat *marker* terdeteksi kemudian terbaca maka akan tampil 9 buah tombol, yaitu tombol *Home, Mute, Zoom In, Zoom Out*, Informasi, *Rotate* Kanan, *Rotate* Kiri, *Rotate* Atas, dan *Rotate* Bawah. Implementasi tampilan halaman *start scan* aplikasi dapat dilihat pada halaman berikutnya. Gambar 5.24 adalah tampilan halaman loading proses start scan.



Gambar 5.24 Halaman Loading *Start Scan*

Tampilan halaman loading akan muncul ketika user menekan tombol start scan pada halaman menu utama. tampilan halaman diatas adalah proses menyiapkan kamera AR untuk men*scan marker*



Gambar 5.25 Tampilan halaman saat kamera mulai mendeteksi *marker*

Setelah *loading*, maka akan muncul halaman seperti gambar diatas. Gambar 5.25 adalah tampilan halaman ketika kamera AR sedang mendeteksi *Marker.* pada saat mendeteksi marker, akan muncul 2 tombol ikon berupa tombol *home* dan tombol *restart*. Setelah *marker* terdeteksi oleh kamera AR, maka akan muncul tampilan seperti pada gambar 5.26 yang dapat dilihat pada halaman berikutnya

Gambar 5.26 Tampilan halaman saat marker terdeteksi oleh kamera

Gambar 5.26 adalah tampilan ketika marker sudah berhasil terdeteksi oleh kamera AR. Ketika marker terdeteksi, maka akan muncul tombol – tombol untuk berinteraksi dengan objek 3D yang muncul dikamera.

Berikut source code halaman *Start Scan.*

a) *Source code* halaman *loading*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
public class Loading : MonoBehaviour
   {
    public GameObject loadingScreen;
    public Slider loadingSlider;
    //public string scenetoload;
public void LoadLevel(int sceneIndex)
       {
StartCoroutine(LoadAsyncronously(sceneIndex));
       }
IEnumerator LoadAsyncronously (int sceneIndex)
      {
AsyncOperation operatioon =
SceneManager.LoadSceneAsync(sceneIndex);
           loadingScreen.SetActive(true);
           while (!operatioon.isDone)
           {
   float progress = Mathf.Clamp01(operatioon.progress /
   .9f);
               loadingSlider.value = progress;
               yield return null;
           }
```

```
        }
    }
```

Berikut penjelasan mengenai source code diatas :

```
    a) public void LoadLevel(int sceneIndex)
        {
            StartCoroutine(LoadAsyncronously(sceneIndex));
        }
       // method untuk melakukan sinkronisasi dengan sceneIndex
    b) {
       AsyncOperation operatioon =
       SceneManager.LoadSceneAsync(sceneIndex);
            loadingScreen.SetActive(true);
            while (!operatioon.isDone)
            {
       float progress = Mathf.Clamp01(operatioon.progress /
    .9f);
       loadingSlider.value = progress;
    // memulai proses sinkronisasi dengan sceneIndex, loading screen dalam keadaan
       aktif. Tampilan proses loading berjalan sesuai / sinkron dengan proses pemuatan
       kamera AR
```

b) Source code halaman mendeteksi *marker*

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System.Collections.Generic;
namespace Vuforia
{
public class targetData : MonoBehaviour
{
public GameObject SemuaButtonController;
// Start is called before the first frame update
    void Start()
    {
    }
        void Update()
        {
      StateManager sm =
      TrackerManager.Instance.GetStateManager();
      IEnumerable<TrackableBehaviour> tbs =
sm.GetActiveTrackableBehaviours();
foreach (TrackableBehaviour tb in tbs)
{
string name = tb.TrackableName;
ImageTarget it = tb.Trackable as ImageTarget;
Vector2 size = it.GetSize();
Debug.Log("Active image target:" + name + "  -size: " + size.x
+ ", " + size.y);
    //Evertime the target found it will show "name of target"
on the TextTargetName. Button, Description and Panel will
visible (active)
```

```
if (name == " ")
{
 SemuaButtonController.gameObject.SetActive(false);
}
else if (name == "")
{
 SemuaButtonController.gameObject.SetActive(false);
}
else
{
SemuaButtonController.gameObject.SetActive(true);
                }
            }
        }
    }
}
```

Berikut penjelasan dari sorce code diatas :

a) **foreach (TrackableBehaviour tb in tbs)**
// membuat perulangan untuk array dengan variabel dibawah
{
string name = tb.TrackableName;
ImageTarget it = tb.Trackable as ImageTarget;
Vector2 size = it.GetSize();
Debug.Log("Active image target:" + name + " -
size: " + size.x + ", " + size.y);
// setiap target marker terdeteksi, maka tombol, deskripsi dan panel akan muncul

b) **SemuaButtonController.gameObject.SetActive(true);**
// membuat semua button controller dalam keadaan active (true)

c) *Source code* halaman saat marker terdeteksi oleh kamera

```
using UnityEngine;
using System.Collections;
using Vuforia;
public class MarkerObjek : MonoBehaviour,
ITrackableEventHandler
{
    private TrackableBehaviour mTrackableBehaviour;
    void Start()
    {
        mTrackableBehaviour =
GetComponent<TrackableBehaviour>();
        if (mTrackableBehaviour)
        {
        mTrackableBehaviour.RegisterTrackableEventHandler(this);
        }
    }
    public void OnTrackableStateChanged(
    TrackableBehaviour.Status previousStatus,
    TrackableBehaviour.Status newStatus)
    {
        if (newStatus == TrackableBehaviour.Status.DETECTED ||
```

```
            newStatus == TrackableBehaviour.Status.TRACKED ||
            newStatus ==
 TrackableBehaviour.Status.EXTENDED_TRACKED)
        {
            for (int i = 1; i <= 8; i++)
            {
GameObject.FindWithTag("TombolUI").transform.GetChild(i).gameObj
ect.active = true;
            }
            foreach (Transform child in transform)
            {
                child.gameObject.active = true;
            }
GameObject.FindWithTag("Model").GetComponent<Animator>().enabled
= true;
            Canvas[] canvasComponents =
 GetComponentsInChildren<Canvas>(true);
            foreach (Canvas component in canvasComponents)
            {
                component.enabled = false;
            }
           // Animator[] animk =
 GetComponentsInChildren<Animator>(true);
          //   foreach (Canvas component in  animk )
          //   {
           //      component.enabled = false;
          //   }
      ///      gameObject.GetComponent<AudioSource>().Play();
          }
          else
          {
            for (int i = 1; i <= 8; i++)
            {
GameObject.FindWithTag("TombolUI").transform.GetChild(i).gameOb
ject.active = false;
            }
            foreach (Transform child in transform)
            {
                child.gameObject.active = false;
            }
          //  gameObject.GetComponent<AudioSource>().Stop();
        }
    }
}
```

Untuk penjelasan source code halaman marker saat terdeteksi dapat dilihat pada halaman berikutnya.

Berikut penjelasan dari source code halaman marker saat terdeteksi :

```
a) newStatus == TrackableBehaviour.Status.DETECTED ||
    // fungsi untuk ketika marker terdeteksi oleh kamera AR
    newStatus == TrackableBehaviour.Status.TRACKED ||
    // fungsi untuk ketika marker berhasil dideteksi oleh kamera AR
newStatus == TrackableBehaviour.Status.EXTENDED_TRACKED)
    // fungsi untuk memperpanjang proses tracking marker

b) GameObject.FindWithTag("TombolUI").transform.GetChild(i).
   gameObject.active = true;
   // memanggil gameObject component dengan nama "TombolUI"

c) GameObject.FindWithTag("Model").GetComponent<Animator>().
   enabled = true;
   // memanggil GameObject component dengan nama "Model"
```

5.3.5 Halaman Soal – Soal

Tampilan halaman soal – soal akan tampil ketika menu soal - soal pada halaman beranda ditekan. Halaman soal - soal berisi pemaparan pertanyaan seputar alat musik daerah. Implementasi tampilan halaman beranda aplikasi dapat dilihat pada Gambar dibawah ini.

Gambar 5.27 Tampilan halaman soal – soal

Pada gambar 5.27 adalah tampilan halaman soal – soal. Halaman diatas akan muncul ketika user menekan tombol soal – soal pada menu utama. soal – soal yang ada dalam menu berisi tentang pertanyaan seputar alat musik daerah di Indonesia. Seperti gambar diatas adalah salah satu contoh soal tentang asal alat musik tersebut.

Gambar 5.28 halaman total nilai

Gambar 5.28 adalah tampilan halaman score akhir dari menjawab pertanyaan pada menu soal – soal. Score akan muncul apabila user telah menjawab pertanyaan yang muncul. Ketika telah selesai, maka user dapat kembali ke halaman utama dengan meneka tombol icon menu utama yang ada di pojok kanan bawah panel.

Berikut source code halaman Soal – soal

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class QuizController : MonoBehaviour
{
    public int question;
    public int score;
    public int addScore;
    public AudioSource quizAudio;
    public AudioClip benar, salah, hasil;
    public Text textResult;
    public GameObject result;
    public GameObject[] qElement;
    private void Update()
    {
        textResult.text = score.ToString();
    }
    public void aTrue(GameObject nextQ)
    {
        next();
        if (question < 5)
        {
            score += addScore;
            trueorfalse(true);
            nextQ.SetActive(true);
        }
        else {
            score += addScore;
```

```
            quizAudio.clip = hasil;
            nextQ.SetActive(true);
            quizAudio.Play();
        }
    }
    public void aFalse(GameObject nextQ)
    {
        next();
        if (question < 5)
        {
            trueorfalse(false);
            nextQ.SetActive(true);
        }
        else {
            quizAudio.clip = hasil;
            nextQ.SetActive(true);
            quizAudio.Play();
        }
    }
    public void home()
    {
        score = 0;
        textResult.text = "";
        for (int i = 0; i < qElement.Length; i++)
        {
            qElement[i].SetActive(false);
        }
        trueorfalse(false);
        question = 0;
    }
    public void next()
    {
        question++;
    }
    void trueorfalse(bool trues)
    {
        if (trues)
        {
            quizAudio.clip = benar;
            quizAudio.Play();
        }
        else {
            quizAudio.clip = salah;
            quizAudio.Play();
        }
    }
}
```

Berikut penjelasan source code diatas :

```
a) textResult.text = score.ToString();
   // menampilkan score hasil menjawab pertanyaan

b) if (question < 5)
     {
      score += addScore;
   // fungsi untuk menghitung score dari hasil menjawab pertanyaan.
```

```
            trueorfalse(true);
              nextQ.SetActive(true);
            }
            quizAudio.Play();
             //  memainkan audio saat next ke pertanyaan selanjutnya
```

5.3.6 Halaman About

Tampilan halaman *about* akan tampil ketika menu about pada halaman beranda ditekan. Halaman *about* berisi informasi identitas pengembang mulai dari nim, nama, dan juga jurusan serta berisi tujuan pembuatan aplikasi. Implementasi tampilan halaman aobout aplikasi dapat dilihat pada Gambar 5.29 berikut ini.

Gambar 5.29 Tampilan halaman *About*

Berikut source code halaman *About.*

```
public void TampilAbout()
    {
     if (GameObject.FindWithTag("About") != null)
    {
     if (about)
    {
GameObject.FindWithTag("About").GetComponent<Canvas>().enabled =
true;
GameObject.FindWithTag("About").transform.GetChild(0).gameObject
.active = false;
GameObject.FindWithTag("About").transform.GetChild(0).gameObject
.active = true;
                about = false;
             }
             else
             {
               //
GameObject.FindWithTag("About").GetComponent<Canvas>().enabled =
false;
```

```
                var animator =
GameObject.FindWithTag("About").transform.GetChild(0).GetCompone
nt<Animator>();
                if
(animator.GetCurrentAnimatorStateInfo(0).IsName("Open"))
                    animator.Play("Close");
StartCoroutine(RunPopupDestroy(GameObject.FindWithTag("About").t
ransform.GetChild(0).gameObject));
                about = true;
            }
        }
    }
```

Berikut penjelasan source code diatas :

a) **`(GameObject.FindWithTag("About")`**
// *class* pada *unity scenes* dengan nama *about*

b) **`GameObject.FindWithTag("About").GetComponent<Canvas>().en abled = true;`**
// menampilkan komponen objek dengan label nama about

c) **`StartCoroutine(RunPopupDestroy(GameObject.FindWithTag("Ab out").transform.GetChild(0).gameObject)); about = true;`**
// memulai proses dengan gameObject tag *about*

5.3.7 Halaman Exit

Tampilan halaman *exit* akan tampil ketika tombol menu *exit* pada halaman utama ditekan. Halaman *exit* berisi panel dengan ucapan kata terima kasih dan tombol ok untuk keluar dari aplikasi. Implementasi tampilan halaman exit aplikasi dapat dilihat pada Gambar 5.30 berikut ini.

Gambar 5.30 tampilan halaman *exit*

Setelah muncul tampilan halaman seperti gambar 5.30, user dapat keluar dari aplikasi dengan menekan tombol ok. Setelah itu akan keluar dari aplikasi disertai backsound *exit* aplikasi.

Berikut source code halaman *exit*

```
public void TampilExit()
    {
      if (GameObject.FindWithTag("Exit") != null)
    {
      if (exit)
    {
GameObject.FindWithTag("Exit").GetComponent<Canvas>().enabled =
true;
GameObject.FindWithTag("Exit").transform.GetChild(0).gameObject.
active = false;
GameObject.FindWithTag("Exit").transform.GetChild(0).gameObject.
active = true;
            exit = false;
            }
            else
            {
            var animator =
GameObject.FindWithTag("Exit").transform.GetChild(0).GetComponen
t<Animator>();
if (animator.GetCurrentAnimatorStateInfo(0).IsName("Open"))
                    animator.Play("Close");
StartCoroutine(RunPopupDestroy(GameObject.FindWithTag("Exit").tr
ansform.GetChild(0).gameObject));
                exit = true;
            }
        }
    }
```

Berikut penjelasan mengenai source code diatas :

a) (GameObject.FindWithTag("Exit")
// *class* pada *unity scenes* dengan nama *exit*

b) GameObject.FindWithTag("Exit").GetComponent<Canvas>().ena bled = true;
// menampilkan komponen objek dengan label nama *exit*

c) StartCoroutine(RunPopupDestroy(GameObject.FindWithTag("Ex it").transform.GetChild(0).gameObject)); exit = true;
// memulai proses dengan gameObject tag *exit*