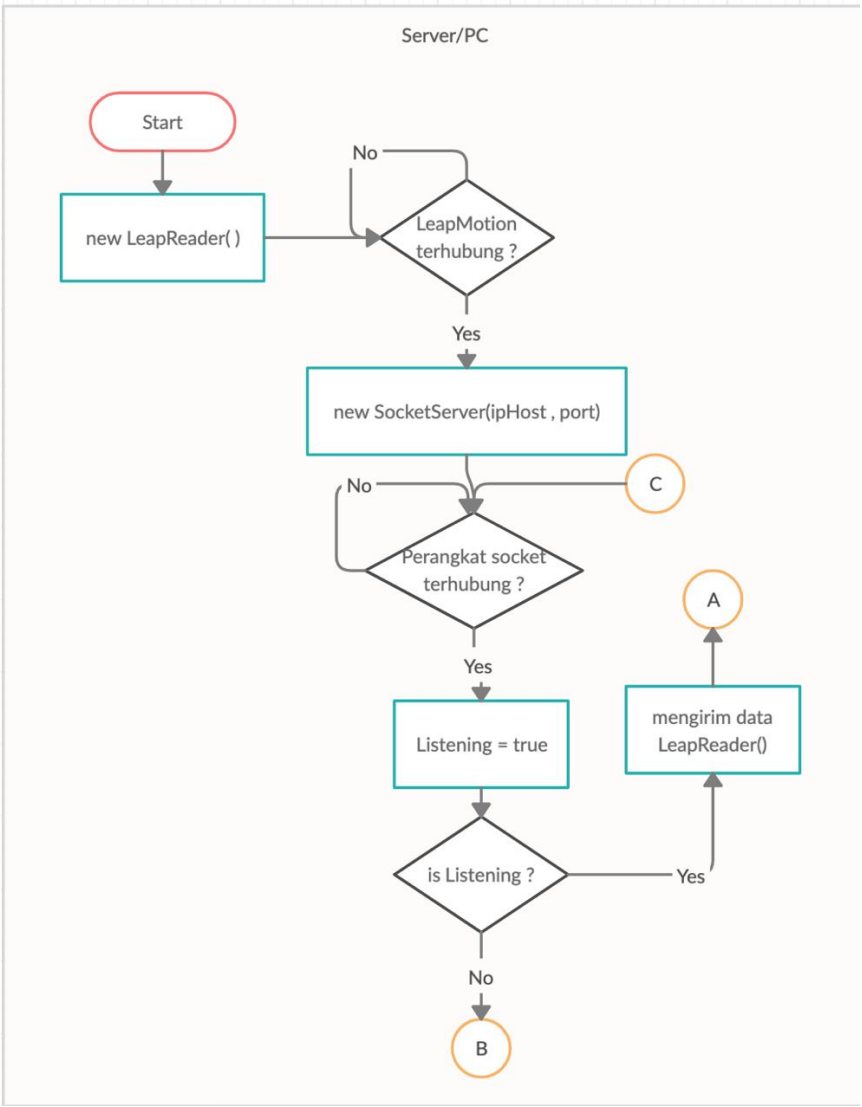
BAB V. IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan tentang pembuatan aplikasi yang sesuai dengan teori yang digunakan dalam pembuatan sistem. Penjelasan berisi langkah-langkah dalam membangun aplikasi. Pada implementasi sistem aplikasi penelitian ini menggunakan bahasa pemrograman C# dalam membuat aplikasi.

5.1 Implementasi Sistem

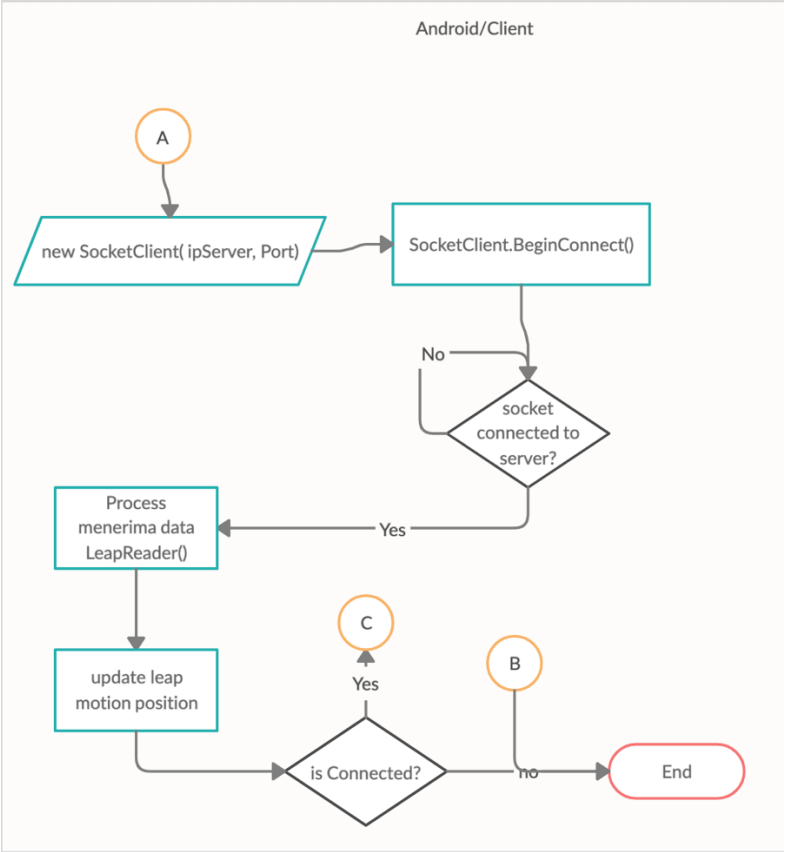
Pada bagian Implementasi sistem merupakan proses pembuatan sistem berdasarkan hasil analisis dan perancangan yang telah dilakukan sebelumnya. Implementasi sistem ini berisi proses bagaimana aplikasi pengenalan hewan nusantara menggunakan *Virtual Reality* dapat terhubung dengan perangkat Leap Motion. Berdasarkan hasil analisis dan perancangan pada bab sebelumnya untuk menghubungkan perangkat Leap Motion dengan perangkat android diperlukan Desktop atau PC, maka hasil implementasi pada penelitian ini akan membuat 2 aplikasi yaitu VR *server* yang berjalan pada Desktop atau PC sebagai *server* dan aplikasi yang kedua yaitu VR *game* belajar mengenal hewan nusantar yang berjalan pada perangkat android yang bekerja sebagai *client*. Aplikasi VR *server* yang berjalan pada Desktop atau PC digunakan sebagai *server* penerima data gerakan tangan yang ditangkap oleh Leap Motion dan kemudian data gerakan tersebut akan dikirim ke perangkat Android melalui jaringan yang sama.

Pada gambar 5.1 merupakan *flowchart* yang digunakan pada aplikasi pada PC, pertama membuat objek baru dari Leap Motion SDK untuk bisa mendeteksi dan membaca data yang terekan pada sensor Leap Motion. Jika perangkat Leap Motion sudah terhubung pada PC, maka dilanjutkan dengan membuat objek Socket baru yang berfungsi sebagai komunikasi antara dua entitas *client* dan *server*. Jika terdapat *client* yang terhubung pada SocketServer dilanjutkan dengan membuat parameter *listening = true*, sebagai indikator perulangan jika *listening == true*? dilanjutkan dengan mengirim data LeapHand yang terekam pada sensor Leap Motion dan jika tidak *listening*, maka SocketServer diberhentikan dan menutup *port* yang sudah terbuka sebelumnya.



Gambar 5.1 *Flowchart* VR *server*

Pada gambar 5.2 merupakan *flowchart* yang terdapat pada android *client*, di awali dengan membuat SocketClient yang digunakan untuk membuat sambungan dengan SocketServer pada IP dan *port* yang digunakan pada SocketServer, kemudian dengan memulai menyambungkan dengan *server* dengan BeginConnect() dan jika *client* sudah terhubung dengan *server*, dilanjutkan dengan menerima data LeapReader dan kemudian melakukan *update* posisi tampilan objek tangan yang akan ditampilkan pada android, jika SocketClient tetap terhubung maka akan tetap terus menerima data dari *server* dan melakukan *update* posisi objek tangan dan jika SocketClient sudah terputus maka sistem berhenti.



Gambar 5.2 *Flowchart* VR android *client*

5.1.1 Sistem VR *Server*

Sistem kerja pada VR *server* ini bekerja dengan mekanisme *Socket Server*, *Socket* adalah sebuah kelas yang digunakan sebagai mekanisme komunikasi untuk pertukaran data antar aplikasi yang terdapat di dalam sebuah mesin maupun beda mesin dan pertukaran ini terjadi pada sebuah jaringan komputer (Rauf et al., 2018). Dalam sistem ini terlebih dahulu menetukan IP dan *port* sebelum membuat koneksi *Socket Server*, terlihat pada *source code* 5.1 dibawah.

Source Code 5.1 konfigurasi Socket Server

```
    void Awake(){
        // Declare Sett LeapMotion Config
        leapReader = new LeapReader(true);
        leapReader.Device += LeapController_Device;
        leapReader.Connect += LeapController_Connect;
        leapReader.DeviceLost += LeapController_DeviceLost;
        leapReader.DeviceFailure+= LeapController_DeviceFailure;
```

```
        // Declare n Config Socket Server
        server = new SocketServer(getLocalIPAddress(), 6321);
        server.OnConnected += Server_OnAddedConnection;
        server.OnDisconnected += Server_OnDisconnected;
        server.OnError += Server_OnError;
        server.SendInterval = sendInterval;
    }
    public void connecting(){
       server.Listen();
       Debug.Log("Started server, waiting for connection...");
    }
```

Setelah mengatur mekanisme komunikasi menggunakan *Socket Server*, kemudian mengambil data dari sensor Leap Motion. LeapReader merupakan class yang berimplementasi dengan *library* Leap.Controller, pada *class* ini akan digunakan untuk menerima dan mengelola data tangan yang terdeteksi pada sensor. Data tangan yang terdeteksi ini akan dikirim dengan mekanisme *Socket Server* yang sudah di tetapkan sebelumnya. Sebelum data tangan dikirim terlebih dahulu data tersebut diolah dan penyesuaian format data dengan yang dibutuhkan, setelah data tersebut diolah kemudian data tersebut dikirim melalui *Socket Server* terlihat pada *source code* 5.2 dibawah.

Source Code 5.2 Mulai Socket *server*

```
public void Listen(){
   if (this.listener == null)
   this.listener   =   new   Socket(AddressFamily.InterNetwork,
   SocketType.Stream, ProtocolType.Tcp);
   this.listenThread = newThread(new ThreadStart(
                                          this.ListenThread))
   { IsBackground = true };
    this.listenThread.Start();
}
```

5.1.2 Sistem VR *Client*

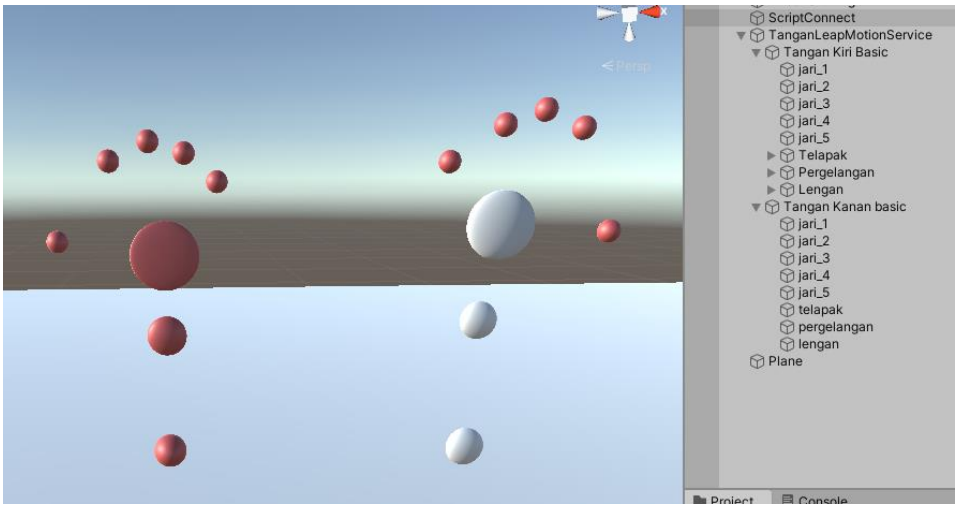
Setelah menelakukan proses pengiriman data pada sub bab sebelumnya, Sistem kerja pada VR *game* pada android yaitu menerima data tangan yang telah dikirim melalui *Socket Server*. Data yang nantinya akan diterima pada perangkat android ini nantinya akan diolah lagi dan ditampilkan dalam VR *environment*. Proses pertama untuk menerima data pada android dengan menghubungkan dengan *server* pada *port* dan IP yang telah ditentukan oleh *Socket Server*, seperti pada *source code* 5.3 dibawah.

Source Code 5.3 Sambungan *Socket Client*

```
public static SocketClient network = new SocketClient();
public static ClientManager ClientMgr;
void Awake () {
    if(ClientMgr == null) ClientMgr = this;
    Leap = new LeapSimulator();
    Leap.AddSource(network); ;
}
void Start(){
   // evnt network socket
   network.OnConnected += (object sender, System.EventArgs e)
             => {
                 Debug.Log("Connected");
                 connected = true; };
   network.OnDisconnected += (object sender,System.EventArgs e)
             => {
                 Debug.Log("Disconnect");
                 connected = false; };
   network.OnError += (object sender, ErrorEventArgs e)
             => {
                 Debug.Log("Error : " + e.Message);
                 connected = false; };
}
void connectToServer(){
   network.Ip = ip;
   network.Port = int.Parse(port);
   network.Connect();
}
public LeapSimulator Leap { get; private set; }
```

Setelah terhubung dengan *Socket server*, pada *source code* 5.3 diatas *class* LeapSimulator digunakan untuk menerima data yang berasal dari *Socket server*. Data yang berhasil diterima kemudian akan ditampilkan pada 3D *environment*.

Sebelum menampilkan pada *environment* terlebih dahulu membuat objek *game* yang akan digunakan sebagai interpretasi dari tangan, karena data yang dikirim oleh *Socket server* berupa nilai posisi dari tangan yang terekam oleh sensor Leap Motion. Gambar 5.3 diabawah merupakan struktur dari objek *game* tangan.



Gambar 5.3 Objek tangan virtual

Objek tangan sudah tersusun selanjutnya melakukan proses penerimaan data dari *Socket server* dan melakukan *update* posisi objek game tangan sesuai dengan data yang dikirim melalui *Socket*. Pada *source code* 5.4 dibawah merupakan isi dari class TanganManager.cs digunakan umtuk melakukan tampilan tangan.

Source Code 5. 4 *Update* tampilan tangan

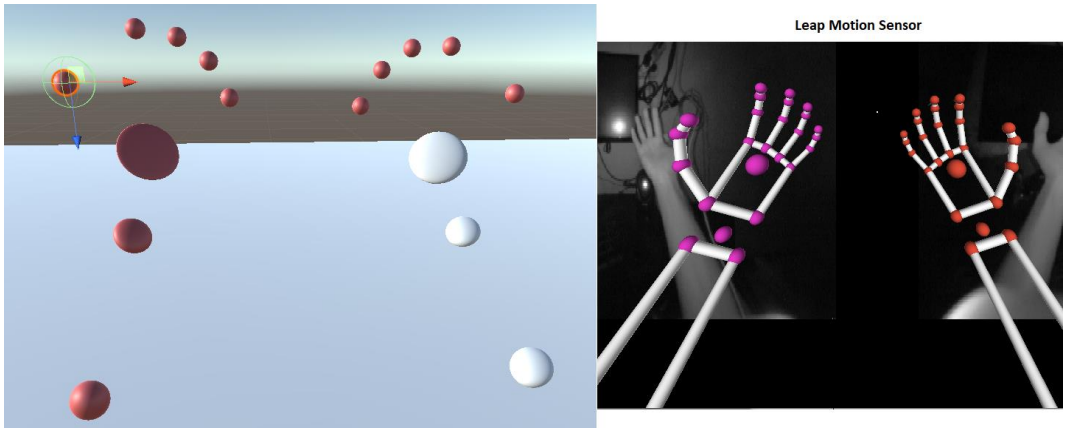
```
ClientManager ClientMgr;
public GameObject tgnKiri;
public GameObject tgnKanan;
TampilanTangan[] tampilTangan;

void Start(){
    ClientMgr = ClientManager.ClientMgr;
    tampilTangan = new TampilanTangan[2];
    tampilTangan[0] = tgnKiri.GetComponent<TampilanTangan>();
    tampilTangan[1] = tgnKanan.GetComponent<TampilanTangan>();
}
void FixedUpdate(){
   if (ClientMgr.Leap.Data != null) {
     List<LeapHand> hands = ClientMgr.Leap.Data.frame.Hands;
     for (int i = 0; i < hands.Count; i++){
         int curSide = hands[i].IsLeft ? 0 : 1;
         hasHandSide[curSide] = true;
         // update posisi tangan
         tampilTangan[curSide].UpdateHand(hands[i]);
     }
   }
}
```

Langkah selanjutnya yaitu melakukan *update* posisi dari objek tangan, langkah ini ada di class TampilanTangan.cs pada fungsi UpdateHand(LeapHand value). Proses dari *update* posisi dapat dilihat dari *source code* 5.5 dibawah.

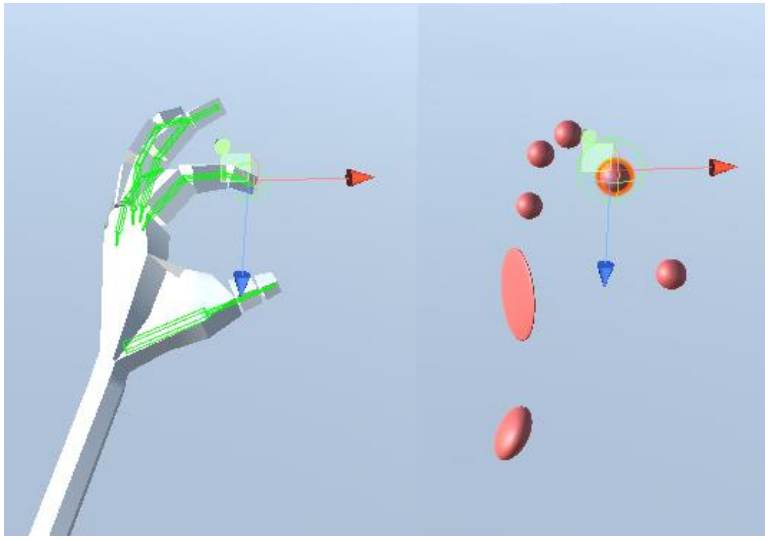
Source Code 5.5 *Update* posisi jari dan tangan

```
public LeapHand tangan;
public Transform[] komposisiTangan;

public void UpdateHand(LeapHand value = null){
   if (value != null) { tangan = value; }
   if (tangan == null) return;

   komposisiTangan[(int)HandPart.Telapak].localPosition =
                  tangan.PalmPosition.ToHMDVector3();
   komposisiTangan[(int)HandPart.Telapak].localRotation =
                  tangan.Rotation.ToHMDQuaternion();
   // posisi jari
   for (int fingerIndex = 0; fingerIndex < tangan.Fingers.Count;
        fingerIndex++) {
      int index = fingerIndex + JARI_PERTAMA_INDEX;
      komposisiTangan[index].localPosition =
         tangan.Fingers[fingerIndex].TipPosition.ToHMDVector3();
   }

   Quaternion armRotation=tangan.Arm.Rotation.ToHMDQuaternion();
   komposisiTangan[(int)HandPart.pergelangan].localPosition =
         tangan.WristPosition.ToHMDVector3();
   komposisiTangan[(int)HandPart.pergelangan].localRotation=
         armRotation;
   komposisiTangan[(int)HandPart.lengan].localPosition =
         tangan.Arm.Center.ToHMDVector3();
   komposisiTangan[(int)HandPart.lengan].localRotation =
         armRotation;
}
```

Hasil dari *source code* 5.5 diatas akan menghasilkan tampilan kurang lebih seperti pada gambar 5.4 dibawah. Dari hasil ini kita sudah bisa melihat tampilan kurang lebih terlihat seperti gambaran tangan, namun agar terlihat seperti tangan yang diperlukan penambahan pergerakan disetiap sendi-sendi tulang. Pada penelitian ini, akan menambahkan metode *invers kinematic*. *Invers Kinematic* merupakan cara untuk mendapatkan besaran sudut dari masing-masing sendi jika diketahui koordinat posisi pangkal dan ujung (Saputro et al., 2018). Hasil penambahan *invers kinematic* dapat dilihat pada gambar 5.5 dibawah.



Gambar 5.4 Implementasi posisi jari tangan



Gambar 5.5 Implementasi *Invers Kinematic* (kiri)

5.2 Implementasi Metode

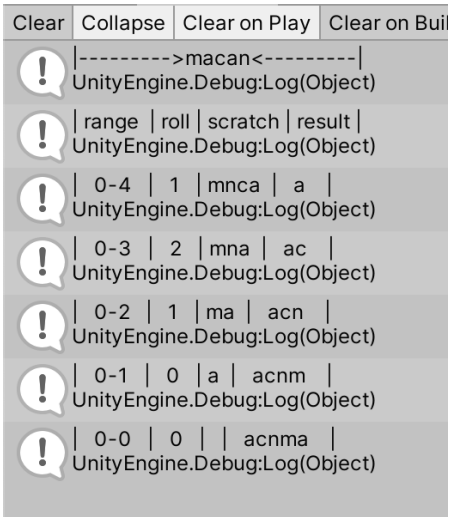
Implementasi metode pada penelitian ini diterapkan pada pengacakan posisi keluaran potongan huruf pada *game*. Berikut pada *source code* 5.6 merupakan implementasi algoritma *Fisher Yates Shuffle* yang di terapkan pada pengacakan nama hewan.

Source Code 5.6 *Fisher Yates Shuffle*

```
char[] fisherYatesSuhuffle(string obj){
    char[] ori = obj.ToCharArray();
    char[] varfys = new char[0];
    //info debug
    Debug.Log("|---->" + new string(ori) + "<----|");
```

```
        Debug.Log("| range  | roll | scratch | result |");
        for (int i = (ori.Length-1); i >=0 ; i--)
        {
            int rand = UnityEngine.Random.Range(0,i);
            varfys = addArray(varfys, ori[rand]);
            ori[rand] = ori[i];
            ori = removeLastArray(ori);
            //
            Debug.Log("| 0-"+i+" | "+rand+" | "+new string(ori)+"
                  "+new string(varfys)+" |" );
        }
        return varfys;
    }

    char[] removeLastArray(char[] val){
        List<char> list = new List<char>(val);
        list.RemoveAt(val.Length - 1);
        return list.ToArray();
    }

    char[] addArray(char[] dst, char x){
        List<char> list = new List<char>(dst);
        list.Add(x);
        return list.ToArray();
    }
```

Hasil dari implementasi *source code* 5.6 diatas menghasilkan pengacakan dari sebuah *string*. Gambar 5.6 dibawah merupakan hasil dari pengacakan menggunakan *Fisher Yates Shuffle* dengan menggunakan *source code* 5.6 dengan menggunakan nama hewan “macan” sebagai masukan nilai *string*, dan menghasilkan nilai acak berupa “acnma”.



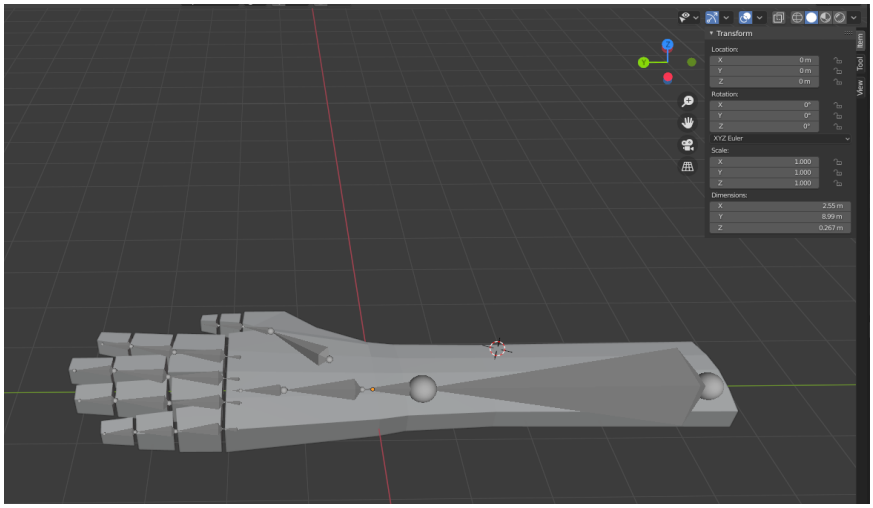
Gambar 5.6 Hasil Pengacakan

5.3 Implementasi Antarmuka

Implementasi antarmuka sistem ini sesuai dengan mockup sistem yang telah dibuat pada bab sebelumnya. Pada implementasi antarmuka VR *game* belajar mengenal hewan nusantara.

5.3.1 Implementasi Objek Tangan

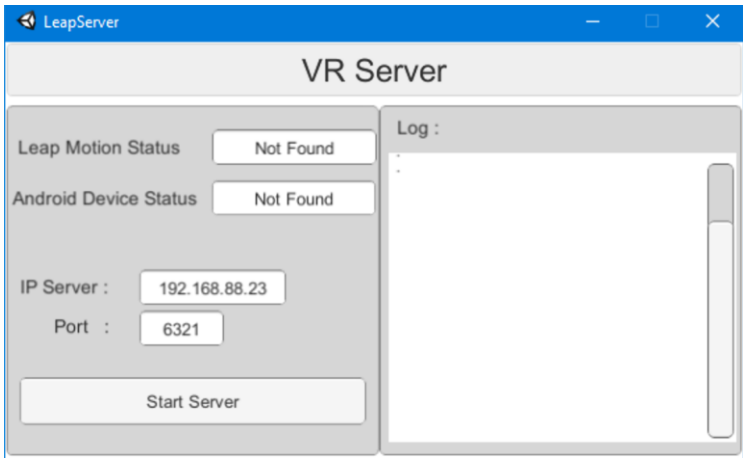
Pada sub bab 5.1.2 sebelumnya telah menghasilkan sebuah objek tangan virtual, akan tetapi tampilan tangan virtual tersebut kurang terlihat seperti tangan sungguhan. Untuk membuat tangan tersebut terlihat semakin baik disini akan membuat 3D objek tangan dengan menggunakan aplikasi Blender, terlihat pada gambar 5.7 dibawah.



Gambar 5.7 3D Objek tangan di Blender

5.3.2 Implementasi VR Server

Gambar 5.8 merupakan tampilan dari aplikasi VR *server* yang dijalankan pada windows. Aplikasi ini wajib dijalankan sebelum menjalankan VR *game* pada perangkat android.



Gambar 5.8 Tampilan aplikasi VR *Server*

5.3.3 Implementasi Sambungan *Server*

Gambar 5.9 dibawah merupakan tampilan sambungan *server*, tampilan ini muncul pertama kali saat menjalankan *game* ini. Pada *scene* ini pengguna diharuskan menginputkan alamat IP dan *port* agar perangkat terhubung dengan Leap Motion dan dapat menjalankan *game*.

Gambar 5.9 Tampilan Sambungan Server

5.3.4 Implementasi Menu Utama

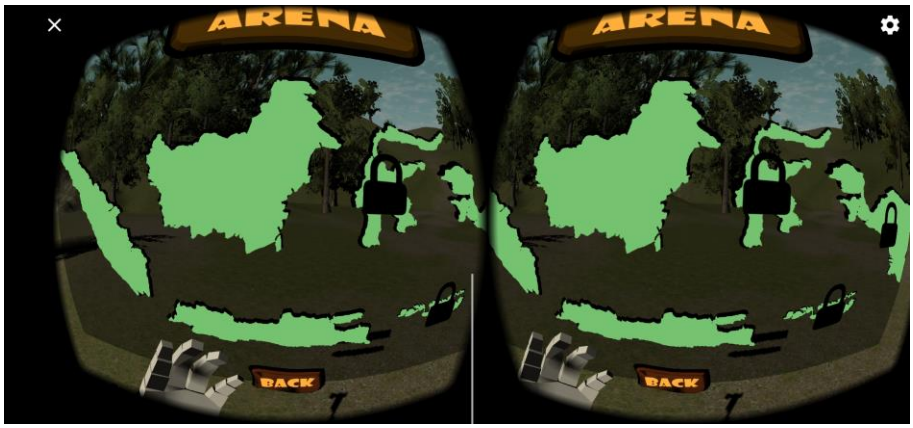
Gambar 5.10 dibawah merupakan tampilan menu utama dari aplikasi belajar mengenal hewan nusantara. Pada halaman ini terdapat tiga pilihan menu yaitu Start

digunakan untuk memulai *game*, Setting digunakan untuk merubah pengaturan suara, dan Quit untuk keluar dari aplikasi ini.

Gambar 5.10 Tampilan menu utama

5.3.5 Implementasi Pilih Pulau

Gambar 5.11 dibawah merupakan tampilan untuk memilih arena pulau. Pada *scene* ini pengguna memilih arena pulau di setiap pulau terdapat hewan endemik masing-masing daerah tersebut.



Gambar 5.11 Tampilan pilih pulau

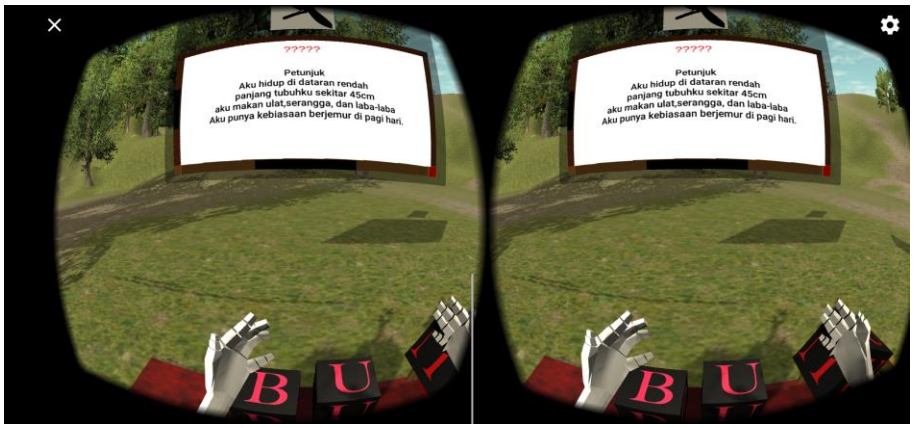
5.3.6 Implementasi Pilih Level

Gambar 5.12 dibawah merupakan tampilan untuk memilih level yang ada pada arena pulau terpilih. Pada *scene* ini pengguna memilih level permainan di pulau tersebut, level permainan ditentukan berdasarkan panjang dari nama hewan yang ada di dalam pulau tersebut.

Gambar 5.12 Tampilan pilih level

5.3.7 Implementasi Permainan

Gambar 5.13 dibawah merupakan tampilan pada saat game dimualai. Pada *scene* ini pengguna menyusun kotak huruf acak menjadi nama hewan yang ada dalam soal.



Gambar 5.13 Tampilan permainan utama

5.4 Pengujian

Pengujian merupakan cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji sistem secara lengkap dan mempunyai kemungkinan untuk menemukan kesalahan. Pengujian pada sistem pada penelitian ini menggunakan 2 pengujian, pengujian pertama menggunakan *Blackbox* dan pengujian UAT (*User Aceptance Test*).

5.4.1 Pengujian *Blackbox*

Pengujian *blackbox* dilakukan untuk mengetahui apakah sistem yang dibangun sesuai dengan kebutuhan yang dirancang sebelumnya. Proses pengujian

blackbox lebih mengarah pada kesesuaian hasil kerja dari sistem pakar dengan kebutuhan yang telah dirancang. Daftar pengujian *blackbox* berdasarkan uji fitur masing-masing kebutuhan akan ditunjukkan pada tabel 5.1 dibawah.

Tabel 5.1 Daftar pengujian *blackbox*

No	Nama uji fitur	Hasil yang diharapkan
1	Input IP & *port*	Dapat menginputkan nilai pada *field* IP & *port*
2	Koneksi ke server	Dapat terhubung dengan *server*
3	Exit	Dapat keluar dari aplikasi
4	Back	Dapat kembali ke *scene* sebelumnya
5	Setting	Dapat masuk ke menu setting
6	Klik tombol	Dapat mengklik tombol dengan menggunakan tangan
7	Memegang box	Dapat memegang dan membawa dengan tangan
8	Pindah level permainan	Dapat pindah ke level berikutnya jika telah menyelesaikan permainan
9	Mengatur ukuran tangan	Dapat menyesuaikan dengan panjang tangan pengguna

5.4.2 Pengujian *User Aceptance Test* (UAT)

Pengujian *User Aceptance Test* merupakan suatu proses pengujian oleh pengguna yang dimaksudkan untuk menghasilkan dokumen yang dijadikan bukti bahwa sistem yang dikembangkan dapat diterima atau tidaknya oleh pengguna, apabila hasil pengujian sudah bisa dianggap memenui maka aplikasi dapat diterapkan (Utomo et al., 2018). Pengujian *User Acceptance Test* menggunakan likert *scale* dengan skala 5. Likert *scale* paling banyak digunakan dalam penelitian yang menggunakan kuisioner kepada responden untuk menentukan tingkat penilaian terhadap kegunaan aplikasi yang telah dibuat (Wibisono Sukmo Wardhono, 2015). Pengujian UAT ini dilakukan dengan mengajukan beberapa pertanyaan terhadap pengguna aplikasi atau anak-anak sekolah dasar yang sebelumnya ditetapkan sebagai target pengguna aplikasi ini, pengujian ini melibatkan 5 anak SD dan 5 masyarakat umum. Daftar pengujian dapat dilihat pada tabel 5.2 dibawah.

Tabel 5.2 Daftar pengujian UAT

No	Aspek Penilaian
Aspek System	
1	Apakah tampilan pada game ini menarik?
2	Apakah tampilan warna dan tampilan pada Media Interaktif ini enak dilihat & tidak membosankan ?
3	Apakah Media Pembelajaran Interaktif mudah dioperasikan..?
Aspek User	
4	Apakah menu-menu pada game media pembelajaran ini menarik dan mudah dipahami ?
5	Apakah materi teka-teki soal mudah di mengerti ?
6	Apakah dengan adanya media pembelajaran ini menambah wawasan tentang hewan nusantara?
Aspek Interaction	
7	Apakah menu dapat diakses dengan mudah?
8	Apakah kontrol dalam aplikasi ini mudah dikendalikan?
9	Apakah semua link dan menu bekerja dengan baik ?

Tabel 5.3 *Likert scale* (Wibisono Sukmo Wardhono, 2015)

Skor Likert	Interpretasi skor interval =20	Pilihan
1	0% - 19.99%	Sangat Tidak Setuju
2	20% - 39.99%	Tidak Setuju

3	40% - 59.99%	Netral
4	60% - 79.99%	Setuju
5	80% - 100%	Sangat Setuju

Tabel 5.4 Bobot Jawaban

Nilai	**Keterangan**	**Bobot**
5	Sangat setuju/bagus/baik	5
4	Bagus/setuju	4
3	Cukup bagus	3
2	Kurang setuju	2
1	Sangat kurang/ buruk	1