# Bab IV

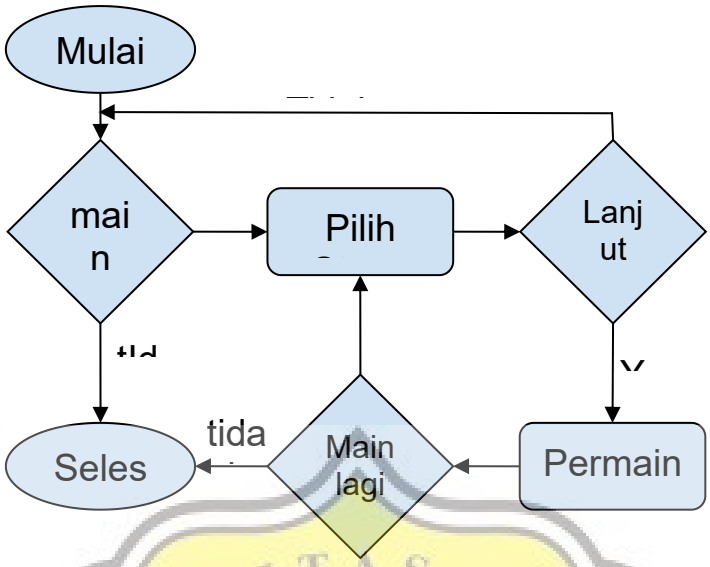
# Perancangan, Pembuatan, dan Pengujian game

## 4.1 Perancangan Game

Game "Nyampah!" dibuat dengan tujuan meningkatkan kesadaran untuk tidak membuang sampah sembarangan dan bisa membedakan(memilah) sampah. Dibuat dengan Unity game ini dibuat tanpa suara karena salah satu input untuk game ini adalah suara. Digunakannya suara agar ada kesan unik dari game ini.

Suara digunakan untuk beberapa input dalam game. Dalam *stage* yang disediakan (ada dua) suara digunakan sebagai input dalam berinteraksi dalam permainan. Penggunaan suara *stage* 1 dan 2 berbeda pada *stage* 1 suara digunakan sebagai input untuk berteriak dan pada *stage* 2 suara digunakan sebagai penentu benar atau tidaknya pernyataan.

Sebelum pembuatan game yang dilakukan terlebih dahulu adalah membuat alur kerja dari game yang akan dibuat. Hal ini dilakukan untuk memudahkan proses pembuatan game karena adanya catatan rinci tentang alur pembuatan sehingga ada garis besar yang akan diikuti dalam membuat game. Alur game dibuat dalam bentuk flowchart gambar 4.1 akan memperlihatkan alur game dari game "Nyampah!"




*Gambar 4.1 Flowchart game Nyampah!*

Tipe permainan dalam game ini ada 2 tipe pertama adalah mengambil sampah(disebut juga *stage* 1).pada *stage* 1 pemain akan bisa berjalan dengan 4 arah akan melihat sampah-sampah bertebaran di sekitar perjalan. Pemain dapat mengambil sampah atau tidak mengambil. Jika sampah diambil maka skor akan bertambah 1 jika tidak diambil skor tidak berkurang sama sekali. Mengambil sampah bisa menggunakan tombol yang telah disediakan di bagian kanan layar.

Pada stage ini juga pemain akan menemui npc berupa anak-anak yang bertugas untuk membuang sampah. Pemain dapat menghentikan

n npc anak tersebut agar tidak jadi membuang sampah dengan mengeluarkan suara. Ketika ada tanda teriak muncul di kepala pemain maka pemain akan mengeluarkan suara agar membuat npc anak tidak membuang sampah sembarangan. Jika pemain terlambat berteriak maka skor akan dikurangi satu jika pemain berhasil menghentikan npc anak tersebut maka skor tidak akan dikurangi.

Pada tahap kedua, game ini akan menjadi seperti kuis. Pada tahap ini sebelum bermain pemain akan memilih jenis sampah yang akan dipilih ke tempat sampah yang disediakan. Ada 4

jenis sampah yang dapat dipilih yaitu sampah gelas, plastik, logam, dan kertas. Walaupun ada 4 sampah yang bisa dipilih ada 1 jenis sampah yang tidak dapat dipilih yang tidak cocok dengan tempat sampah apapun yang disediakan. Sampah ini disebut dengan sampah berbahaya.

Ketika permainan dimulai maka pemain akan memilih jenis sampah yang sama dengan tempat sampah yang ada. Ketika pemain melihat jenis sampah yang berbeda dengan tempat sampahnya maka pemain akan mengeluarkan suara sebelum waktu yang habis. Jika sampah yang diberikan sama maka pemain hanya menunggu waktu habis tanpa mengeluarkan suara.

### 4.2 Pembuatan Game

Game "Nyampah!" Dibuat Dengan menggunakan *Unity* menggunakan Bahasa pemrograman C#. *unity* dipakai untuk membuat dan menyatukan semua aset yang telah disediakan atau dibuat di aplikasi menggambar yang lain.

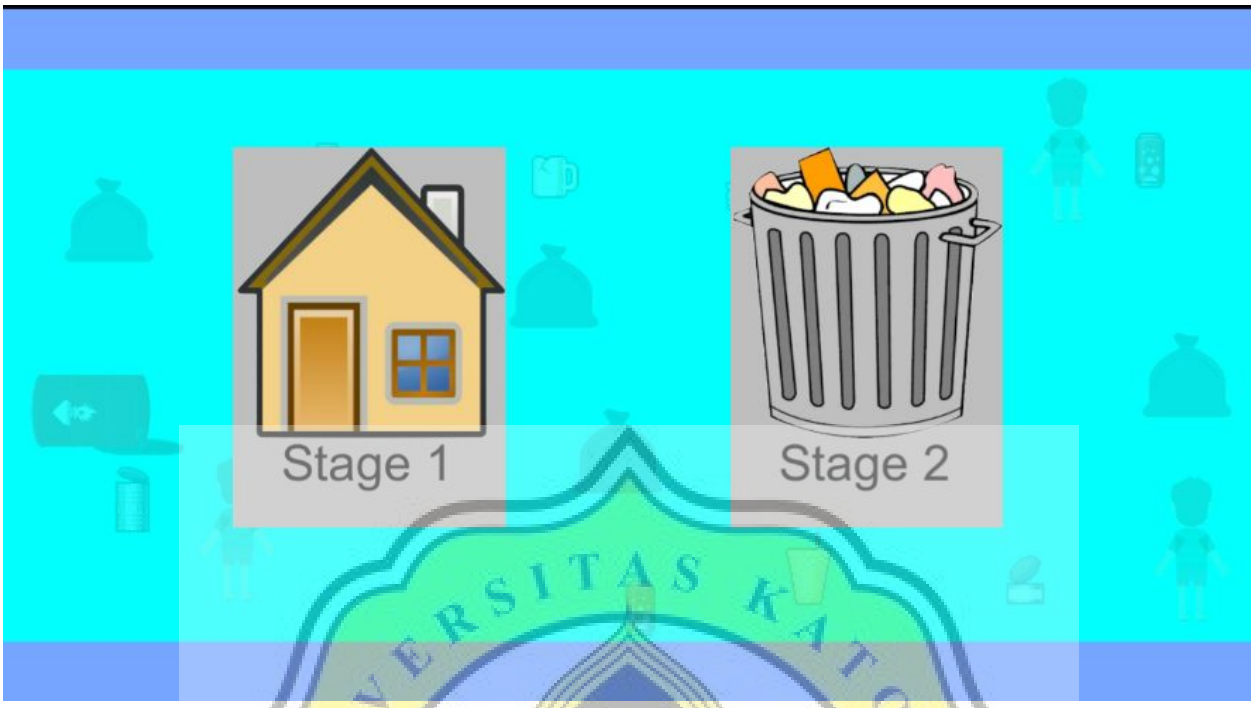
#### 4.2.1 Main Menu

Untuk memulai game paling awal dibuat *main menu*. *Main menu* berisi Tombol seperti main dan exit. Bisa juga berisi hal lain seperti pengaturan. Tombol ini akan memiliki fungsinya sendiri-sendiri. Gambar 4.2 akan memperlihatkan tampilan dari *main menu*.


*Gambar 4.2 Tampilan Main menu*

### 4.2.2 Pilih Stage

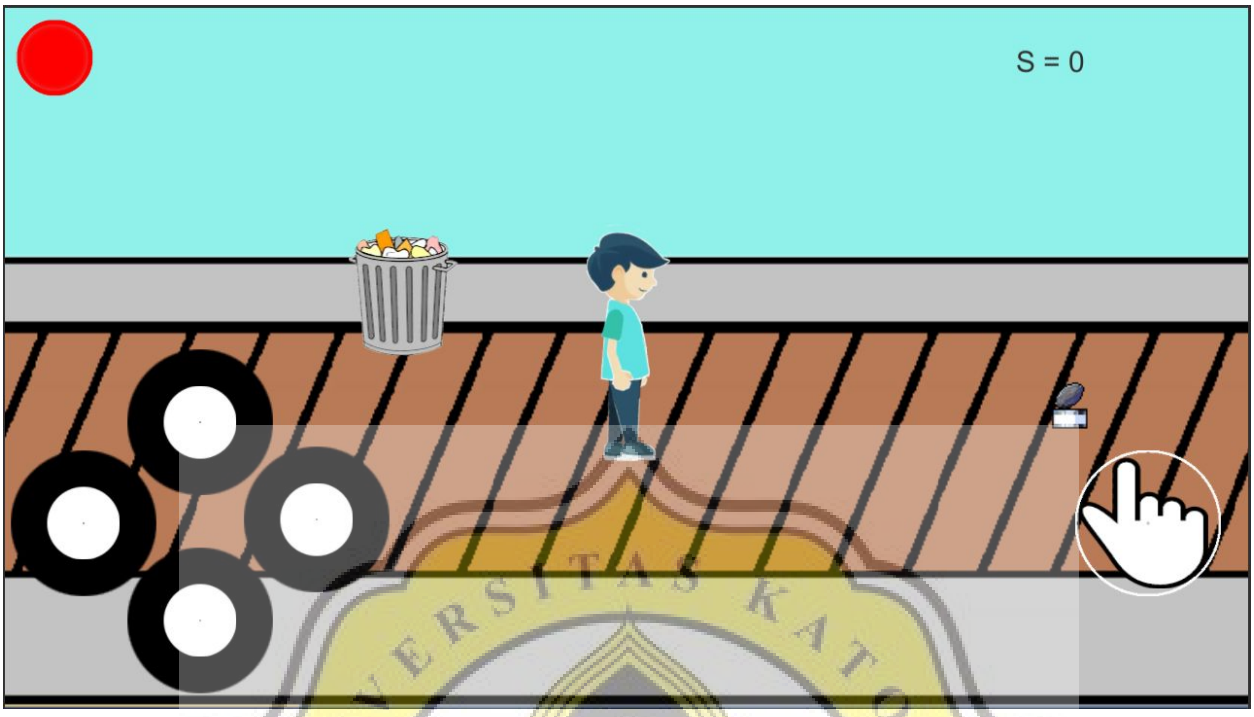
Ketika pemain memilih main maka pemain akan memilih stage yang akan dimainkan. Pada tahap ini pemain akan memilih 1 dari 2 stage yang tersedia gambar 4.3 akan menunjukkan tampilan ketika memilih stage.




*Gambar 4.3 memilih stage*

### 4.2.3 Stage 1

Ketika pemain memilih stage 1 maka pemain akan berpindah scene ke scene yang mengatur stage 1.gambar 4.4 dan 4.5 akan menunjukkan stage yang akan dipilih. Stage akan terpilih secara acak.




*Gambar 4.4 Tampilan stage 1 pilihan 1*


*Gambar 4.5 Tampilan stage 1 pilihan 2*

Kode berikut akan menunjukkan kode untuk memilih stage

```
public class menu : MonoBehaviour
{
    // Start is called before the first frame update
```

```
    public void stage1()
    {
        SceneManager.LoadScene(Random.Range(1, 3));
    }

    public void stage2()
    {
        SceneManager.LoadScene(3);
    }
}
```

### 4.2.4 Kamera

Pada stage 1 kamera akan mengikuti pemain dengan syarat tertentu stage akan mengikuti pemain mengikuti gari x dan y kode di bawah ini akan menunjukkan *coding* yang dipakai untuk mengikuti player

```
public class camera : MonoBehaviour
{
    // Start is called before the first frame update
    public Transform pla;
    public float f,c,y;
    public Vector3 offset;

    void Start()
    {
        f = pla.position.y;
        c = pla.position.x;
    }
    // Update is called once per frame
    void FixedUpdate()
    {

        y = f - pla.position.y;
        //Debug.Log(y + " f");

        if (c > 20)
        {
            Debug.Log("jalan");
        }

        else if (pla.position.x > c)
        {
            transform.position = new Vector3(c, transform.position.y) + offset;
            c = pla.position.x;
        }
```

```
        if( pla.position.y > -4.90f && pla.position.y < -1.19f)
        { transform.position = new Vector3(transform.position.x, -y) + offset; }
    }
}
```

### 4.2.5 Sampah Berserakan di jalan

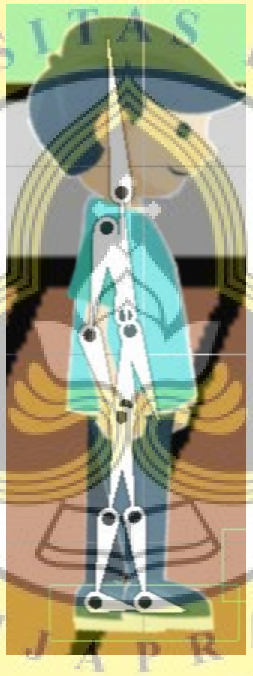
Pada stage 1 sampah akan bermunculan di jalan secara acak dan player disuruh untuk mengambil sampah yang bertebaran di jalan. jenis sampah yang bertebaran akan muncul secara acak. Sampah akan muncul ketika pemain sudah berjalan sebanyak jarak yang ditentukan dan sampah akan muncul di jalan secara acak. Dibawah ini merupakan contoh dari kode yang digunakan untuk menampilkan sampah

```
public class spawnsampah : MonoBehaviour
{
    // Start is called before the first frame update
    public GameObject[] sampah;
    public Transform pla;
    float pos, pos2 ,jarak ;
    bool spawn =false;
    GameObject aws;
    int sp;
    void Start()
    {
        jarak = 15;
        sp = 1;
    }

    // Update is called once per frame
    void Update()
    { pos = pla.position.x;
        pos2 = pos % jarak;
        if (pos2 > 0 && pos2 < 0.1f && spawn)
        {
            if (jarak >= 4 && sp == 1)
            {
                aws = Instantiate(sampah[Random.Range(0, 3)], new Vector3(pla.position.x + Random.Range(-2, 2),
Random.Range(-2.75f, 0), transform.position.z), pla.rotation);
                aws.transform.localScale = new Vector2(2.2f, 2.2f);
                aws.AddComponent<BoxCollider2D>();
```

```
            aws.GetComponent<BoxCollider2D>().size = new Vector2(.3f, .4f);
            aws.GetComponent<BoxCollider2D>().isTrigger = true;
            jarak = jarak - 0.5f;
            spawn = !spawn;
        }

        else if (jarak < 4 && sp == 1)
        {
            aws = Instantiate(sampah[Random.Range(0, 3)], new Vector3(pla.position.x + Random.Range(-2, 2),
Random.Range(-2.75f, 0), transform.position.z), pla.rotation);
            aws.transform.localScale = new Vector2(2.2f, 2.2f);
            aws.AddComponent<BoxCollider2D>();
            aws.GetComponent<BoxCollider2D>().size = new Vector2(.3f, .4f);
            aws.GetComponent<BoxCollider2D>().isTrigger = true;
            sp = 2;
            jarak = 15;
            spawn = !spawn;
        }

        else if (jarak >= 4 && sp == 2)
        {
            aws = Instantiate(sampah[Random.Range(0, 3)], new Vector3(pla.position.x + Random.Range(-2, 2),
Random.Range(-2.75f, 0), transform.position.z), pla.rotation);
            aws.transform.localScale = new Vector2(2.2f, 2.2f);
            aws.AddComponent<BoxCollider2D>();
            aws.GetComponent<BoxCollider2D>().size = new Vector2(.3f, .4f);
            aws.GetComponent<BoxCollider2D>().isTrigger = true;

            aws = Instantiate(sampah[Random.Range(0, 3)], new Vector3(pla.position.x + Random.Range(-2, 2),
Random.Range(-2.75f, 0), transform.position.z), pla.rotation);
            aws.transform.localScale = new Vector2(2.2f, 2.2f);
            aws.AddComponent<BoxCollider2D>();
            aws.GetComponent<BoxCollider2D>().size = new Vector2(.3f, .4f);
            aws.GetComponent<BoxCollider2D>().isTrigger = true;
            sp = 2;
            jarak = jarak - 0.5f;
            spawn = !spawn;
        }

        else if (jarak <= 4 && sp == 2)
        {
            selesai();
        }
    }
    else if (!spawn && pos2 >= 0.1f)
    {
        spawn = !spawn;
    }
  }

  void selesai()
  { }
}
```

### 4.2.6 Player

Pada stage 1 akan ada player yang dapat digunakan oleh pemain. Player ini akan digunakan untuk mengambil sampah yang berserakan di jalan dan berteriak kepada anak-anak yang akan membuang sampah sembarangan. Gambar 4.6 akan menunjukkan gambar pemain yang akan dipakai.



*Gambar 4.6 Player yang digunakan*

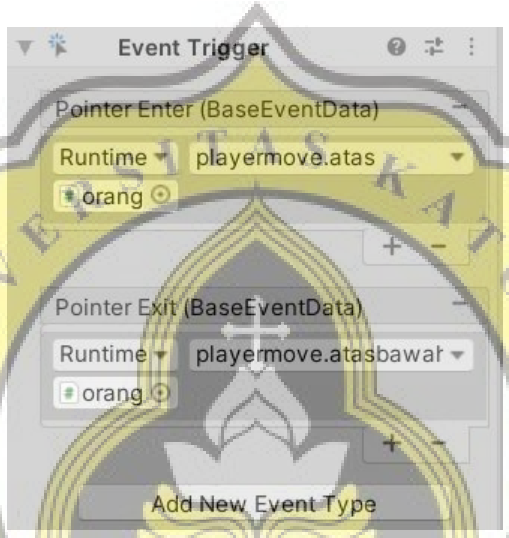
### 4.2.7 Tombol gerak dan ambil sampah Player

Pada stage 1 player akan digerakkan dengan tombol. Tombol akan berguna untuk menjalankan player dan mengambil sampah. Berikut adalah script untuk menjalankan player

```
 else if (math.abs(h) > 0 && getbisagerak & !getteriak && !getjongkok || math.abs(v) >0 && getbisagerak && !getteriak && !getjongkok)
        {
```

```
        gameObject.transform.position = new UnityEngine.Vector2(transform.position.x + (h * speed),transform.position.y + (v * speed));
        setdiem = 0;

    }
```

Variabel h akan diatur dari tombol yang ada di layar untuk menjalankan dan masing-masing tombol akan menjalankan fungsinya masing-masing gambar 4.7 berikut akan menunjukkan fungsi salah satu tombol



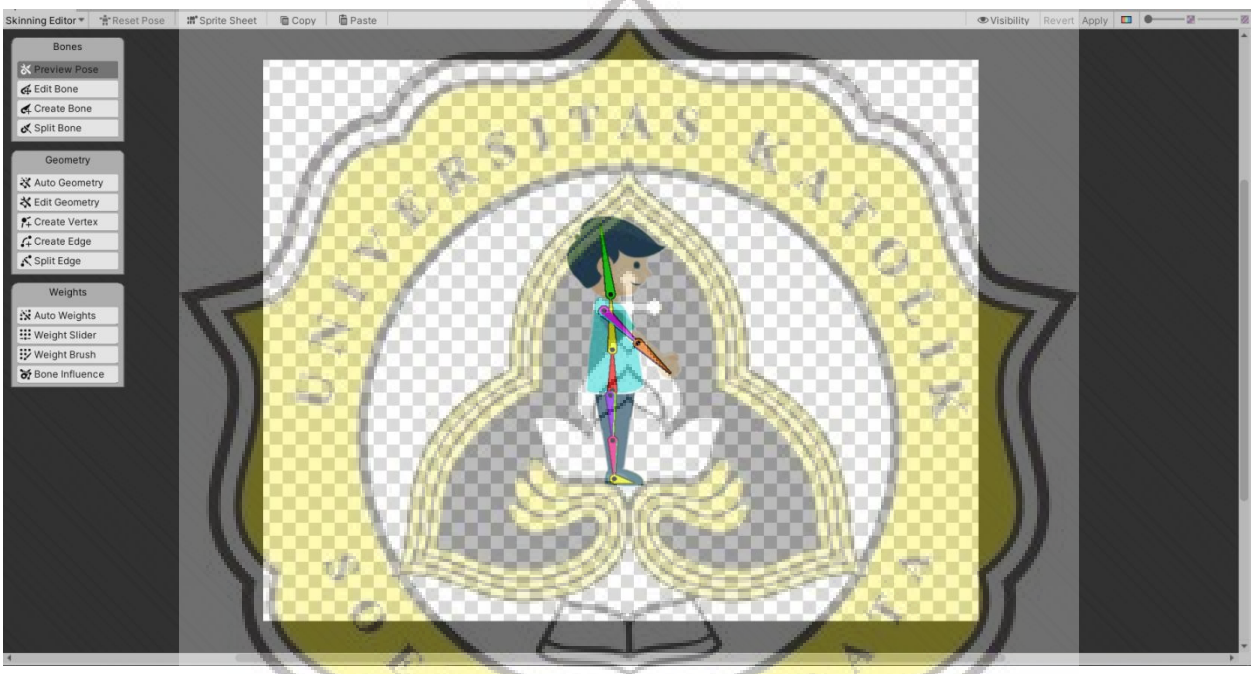

*Gambar 4.7 menunjukkan salah satu fungsi tombol*

Setiap fungsi yang diberikan di tombol mempunyai kode sederhana. Berikut contoh kode pada salah satu tombol gerak

```
public void atas()
   {
      v = 1;
   }

  public void atasbawah()
   {
      v = 0;
   }
```

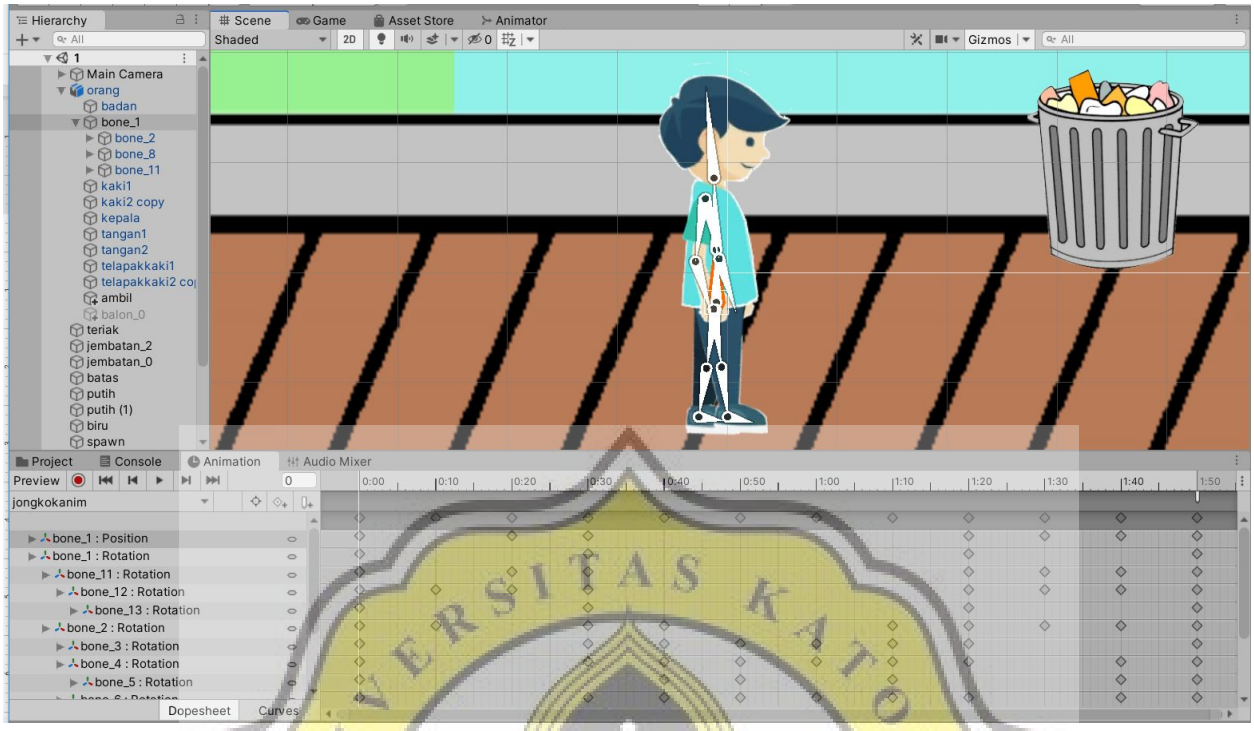
### 4.2.8 Animasi Player

Pada stage 1 player membutuhkan animasi untuk bergerak, mengambil sampah, dan berteriak. Animasi dibuat di dalam aplikasi unity menggunakan *package manager 2D animation*. *2D animation* digunakan untuk membuat bone pada gambar sehingga memudahkan membuat animasi. Gambar 4.8 berikut akan menunjukkan *skinning editor module* dari *package 2D animation.*



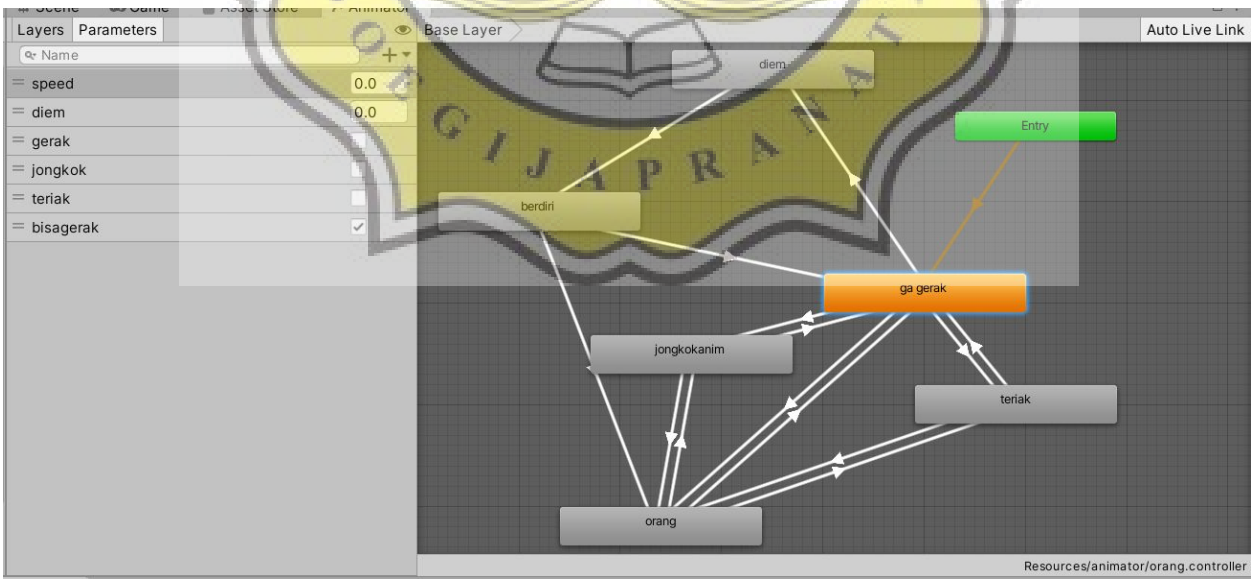

*Gambar 4.8 tampilan skinning editor*

Setelah *bone* selesai diatur dan sudah sesuai dengan gambar maka dilanjutkan dengan membuat animasi. Membuat animasi dilakukan di unity. Aplikasi unity sudah menyediakan tempat untuk membuat animasi gambar 4.9 akan menunjukkan tampilan ketika membuat animasi




*Gambar 4.9 tampilan ketika membuat animasi*

Setelah animasi dibuat maka akan dibutuhkan parameter untuk mengatur kapan animasi akan keluar contoh gambar 4.10 akan menunjukkan tampilan ketika kita mengubah parameter




*Gambar 4.10 tampilan animator tempat mengubah parameter*

Setelah pengaturan parameter maka akan dibuat kode untuk mengatur animasi dan parameter agar animasi berjalan kode dibawah ini merupakan contoh penggunaan parameter pada *script* agar animasi berjalan.

```
void Start()
  {
     //ambil value awal dari animation untuk dijadikan patokan
     setduduk = animator.GetBool("gerak");
     setjongkok = animator.GetBool("jongkok");
     setteriak = animator.GetBool("teriak");
     setbisagerak = animator.GetBool("bisagerak");
     setspeed = animator.GetFloat("speed");
     setdiem = animator.GetFloat("diem");
     over.SetActive(false);
     kontrol.SetActive(true);

  }

  // Update is called once per frame
  void Update()
  {
     sm.text = "S = " + score;

     Debug.Log(ambik);
     //ngatur value animasi yang sudah diupdate
     animator.SetBool("gerak", setduduk);
     animator.SetBool("jongkok", setjongkok);
     animator.SetBool("teriak", setteriak);
     animator.SetBool("bisagerak", setbisagerak);
     animator.SetFloat("speed", setspeed);
     animator.SetFloat("diem", setdiem);



     //ngambil value animasi yang diupdate
     getduduk = animator.GetBool("gerak");
     getjongkok = animator.GetBool("jongkok");
     getteriak = animator.GetBool("teriak");
     getbisagerak = animator.GetBool("bisagerak");
     getspeed = animator.GetFloat("speed");
     getdiem = animator.GetFloat("diem");


     //ngatur pergerakan
     /* h = Input.GetAxisRaw("Horizontal");
     v = Input.GetAxisRaw("Vertical");*/
     setspeed = Mathf.Abs(h) + Mathf.Abs(v);
```

```
    if (h>0 && !liatkanan && getbisagerak && !getteriak ||h<0 && liatkanan && getbisagerak && !getteriak)
    {

       Flip();
    }


}

void FixedUpdate()
{



   // Debug.Log(setjongkok + " jongkok");
    if (getspeed == 0 && !getduduk)
    {
       setdiem += Time.deltaTime;
       if (setdiem >= 5)
       { setduduk = !setduduk;
       }
    }

     if (math.abs(h) > 0 && getduduk || math.abs(v)> 0 && getduduk)
     {
        setduduk = !setduduk;
        setdiem = 0;
     }


      else if (math.abs(h) > 0 && getbisagerak & !getteriak && !getjongkok || math.abs(v) >0 && getbisagerak
&& !getteriak && !getjongkok)
      {
         gameObject.transform.position = new UnityEngine.Vector2(transform.position.x + (h *
speed),transform.position.y + (v * speed));
         setdiem = 0;

      }


}




public void bisagerakf()
{
   setbisagerak = !setbisagerak;
```

```
}

public void teriakf()
{
   setteriak= !setteriak;
}

public void jongkokf()
{
   setjongkok = !setjongkok;
}
void Flip()
{
   // Switch the way the player is labelled as facing
   liatkanan = !liatkanan;
   flips = !flips;

   // Multiply the player's x local scale by -1
   UnityEngine.Vector3 theScale = transform.localScale;
   theScale.x *= -1;
   transform.localScale = theScale;
}

public void teriakanim()

{
   { setteriak = !setteriak; }
}


public void OnTriggerEnter2D(Collider2D collision)
{
   if (collision.gameObject.tag == "Finish")
   {
      over.SetActive(true);
      kontrol.SetActive(false);
   }
}

public void OnTriggerStay2D(Collider2D collider)
{
   if (collider.gameObject.tag == "pikup")
   {
      if (  !setjongkok)
      { if (ambik == 1)
         {
            setjongkok = !setjongkok;
            Destroy(collider.gameObject);
            score += 1;
            sampah_ambil += 1;
         }
      }
```

```
    }
}
```

Kode diatas mengatur animasi yang digabung ke dalam pergerakan player jadi *script* tersebut dipakai pada satu objek yaitu player untuk mengatur pergerakan

### 4.2.9 Player mengambil Sampah

Pada stage 1 player diharuskan mengambil sampah agar skor menambah pemain bisa mengambil sampah jika sampah berada di dekat player. Kode di bawah ini akan menunjukkan kode yang digunakan untuk mengambil sampah

```
public void OnTriggerStay2D(Collider2D collider)
 {
    if (collider.gameObject.tag == "pikup")
    {
       if (  !setjongkok)
       { if (ambik == 1)
          {
             setjongkok = !setjongkok;
             Destroy(collider.gameObject);
             score += 1;
             sampah_ambil += 1;
          }
       }

    }

}
```

### 4.2.10 Player berteriak ketika pemain mengeluarkan suara

Player akan bertemu NPC anak yang akan membuang sampah sembarangan. Pemain akan diminta untuk berteriak ketika berada di dekat anak agar anak tidak jadi membuang sampah sembarangan. Kode berikut ini akan menunjukkan kode yang membuat suara dapat diterima dan diubah menjadi satuan suara agar dapat dijadikan syarat untuk keras suara yang diminta

```
public class sound : MonoBehaviour {
   public float semsi = 100;
   public float loud = 0;
   public Text aaa;
   AudioSource _audio;
   // Use this for initialization
   void Start() {
      _audio = GetComponent<AudioSource>();
      _audio.clip = Microphone.Start(null, true, 10, 44100);
      _audio.loop = true;
      _audio.mute = false;
      while (!(Microphone.GetPosition(null) > 0)){}
      _audio.Play();
          }

          // Update is called once per frame
          void Update () {
      loud = getv() * semsi;
      aaa.text ="Skor = " + gameObject.GetComponent<gamecontroller>().score.ToString();
   }
   float getv()
   {
      float[] data = new float[256];
      float a = 0;
      _audio.GetOutputData(data, 0);
      foreach (float s in data)
      {
         a += Mathf.Abs(s);
      }
      return a / 256;
   }
}
```

Setelah kode dibuat untuk membuat input suara memungkinkan maka dibuat kode untuk membuat anak teriak ketika dekat dengan anak yang ingin membuang sampah sembarangan agar kode berjalan. Kode berikut akan menunjukkan kode agar anak berteriak ketika pemain bersuara dengan tingkat keras tertentu

```
public class teriak : MonoBehaviour
{
   // Start is called before the first frame update
   public Transform pla;
   public float f, c, y, h;
   public Vector3 offset;
   bool s = true;
   public GameObject bub;
   [HideInInspector]
   public int bisateriak = 0;
```

```
public bool balonmuncul = false;

void Start()
{
    f = pla.position.y;

}
// Update is called once per frame
void FixedUpdate()
{

    h = pla.gameObject.GetComponent<playermove>().h;
    if (h > 0 && !s || h < 0 && s)
    {
        Debug.Log("jalan");
        muter();
    }


    c = pla.position.x;
    y = f - pla.position.y ;
   // Debug.Log(y + " f");
    if (c >= 0)
        transform.position = new Vector3(c, transform.position.y) + offset;
    if( pla.position.y > -4.90f && pla.position.y < -1.19f)
    { transform.position = new Vector3(transform.position.x, -y) + offset; }



    Debug.Log(s + " s bool");
    Debug.Log(h + " h float");
}

void muter()
{
    s = !s;

    // Multiply the player's x local scale by -1
    Vector3 theScale = transform.localScale;
    theScale.x *= -1;
    transform.localScale = theScale;

}

void OnTriggerEnter2D(Collider2D collider)
{


    if (collider.gameObject.tag == "teriak" && !balonmuncul)
    {
        bisateriak = 1;
        bub.SetActive(true);
        balonmuncul = !balonmuncul;
```

```
        }

        else if (collider.gameObject.tag == "teriak")
        {

        }
    }

    void OnTriggerStay2D(Collider2D collision)
    {
        if (pla.gameObject.GetComponent<playermove>().getteriak == false)
        {
            if (gameObject.GetComponent<soundcopy>().loud > 5)
            {
                pla.gameObject.GetComponent<playermove>().teriakanim();
                bisateriak = 2;
                bub.SetActive(false);
            }
        }
    }

    void OnTriggerExit2D(Collider2D collider)
    {
        if (collider.gameObject.tag == "teriak") { bub.SetActive(false); }


    }

}
```

### 4.2.11 NPC anak

Anak pada stage 1 akan muncul di tempat yang sudah ditentukan berikut kode untuk memunculkan anak pada tempat-tempat yang sudah ditentukan

```
public class spawnanak : MonoBehaviour
{
    // Start is called before the first frame update
    public List <GameObject> a;
    GameObject ba, sa;
    public Transform pla;
    public GameObject anak,teriak;
    Animator anima;
    int b = 0;
    float anaka,anakb,jarak ;
    public float asa;
```

```
bool spawn =false;
GameObject aws;
void Start()
{
   anaka = 0;
   jarak = 20;
}

// Update is called once per frame
void Update()
{
   anakb = transform.position.x;


   if (jarak > 8)
   {
      if (anaka - anakb < -jarak)
      {
         spawnbaru();
      }
   }
  /*else if (anaka - anakb < -30)
   { Destroy(a); }*/
}


   public void spawnbaru()
{
   ba = Instantiate(anak, new Vector3(transform.position.x + 1, anak.transform.position.y +asa,
anak.transform.position.z), transform.rotation);
   ba.transform.parent = GameObject.Find("spawn_anak").transform;
   ba.name = "anak_" + b;
   ba.AddComponent<Animator>();
   anima = ba.GetComponent<Animator>();
   anima.runtimeAnimatorController                                                    =
(RuntimeAnimatorController)Instantiate(Resources.Load("anaksampah"));


   anak.transform.position = new Vector3(transform.position.x + 1, anak.transform.position.y,
transform.position.z);
   anaka = anak.transform.position.x;
   teriak.gameObject.GetComponent<teriak>().balonmuncul = false;
   anak.gameObject.GetComponent<ator_anim>().blombuang = true;
   teriak.gameObject.GetComponent<teriak>().bisateriak = 0;
   jarak = jarak - 1;
}

void OnTriggerEnter2D(Collider2D collider)
{
   if (collider.gameObject.tag == "teriak")
   {
      Debug.Log("ntaps");
      Destroy(collider.gameObject);
```

```
    }

  }
}
```

### 4.2.12 NPC Anak buang sampah termasuk animasi

Anak pada stage 1 bertugas untuk membuang sampah sembarangan. Sebagai player pemain diminta untuk berteriak ketika melihat anak yang akan membuang sampah. Jika pemain terlalu lama berteriak atau tidak teriak sama sekali maka anak akan membuang sampah jika pemain berteriak sebelum anak membuang sampah maka anak tidak jadi membuang sampah. Berikut ini akan ditunjukkan kode yang dipakai untuk animasi anak

```
public class ator_anim : MonoBehaviour
{

  private Animator animator;
  GameObject orang, teriak, balon;
  private int setanim, getanim;
  public bool blombuang =true;
  int bisateriak;
  // Start is called before the first frame update
  void Start()
  {
    animator = gameObject.GetComponent<Animator>();
    setanim = animator.GetInteger("ambil");
    /*setteriak = animator.GetBool("teriak");
    setbisagerak = animator.GetBool("bisagerak");
    setspeed = animator.GetFloat("speed");
    setdiem = animator.GetFloat("diem");
   //  orang = ;*/
    orang = GameObject.Find("orang");
    teriak = GameObject.Find("teriak");
    balon = GameObject.Find("/orang/balon_0");
  }

  // Update is called once per frame
  void Update()
  {
    /*if (blombuang)
    {
      bisateriak = teriak.GetComponent<teriak>().bisateriak;
```

```
    }
    else if (!blombuang)
    { }*/


    animator.SetInteger("ambil", bisateriak);
    animator.SetBool("adasampah", blombuang);
    /*animator.SetBool("teriak", setteriak);
    animator.SetBool("bisagerak", setbisagerak);
    animator.SetFloat("speed", setspeed);
    animator.SetFloat("diem", setdiem);
    */

    getanim = animator.GetInteger("ambil");
    /*getjongkok = animator.GetBool("jongkok");
    getteriak = animator.GetBool("teriak");
    getbisagerak = animator.GetBool("bisagerak");
    getspeed = animator.GetFloat("speed");
    getdiem = animator.GetFloat("diem");*/
}

public void gantianim()
{
    bisateriak = 2;
}

public void terlambat()
{
    blombuang = !blombuang;

}

public void gantiawal()

{
    bisateriak = 0;
    orang.GetComponent<playermove>().score -= 1;
    orang.GetComponent<playermove>().sampah_buang = 1;
}

void OnTriggerEnter2D(Collider2D collider)
{


    if (collider.gameObject.tag == "Player")
    {
        bisateriak = 1;
    }

}

void OnTriggerStay2D(Collider2D collision)
```

```
    {

        /* if (pla.gameObject.GetComponent<playermove>().getteriak == false)
         {
            if (gameObject.GetComponent<soundcopy>().loud > 5)
            {
               pla.gameObject.GetComponent<playermove>().teriakanim();
               bisateriak = 2;
               bub.SetActive(false);
            }
         }*/

        if (collision.gameObject.tag == "Player")
        {
           if (teriak.GetComponent<teriak>().bisateriak == 2 && teriak.GetComponent<soundcopy>().loud >5)
           {
              bisateriak = 2;
           }
        }


    }
}
```

### 4.2.13 Background Stage 1

Di stage ini background dibuat agar bisa berpindah-pindah sehingga tidak perlu pembuatan background yang terlalu banyak background yang dibuat hanya sedikit saja. Berikut contoh kode yang digunakan

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class pindah : MonoBehaviour
{
    public GameObject sas;
    public float jarak, jarak2;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
```

```
    {
        if (transform.position.x - sas.transform.position.x < -jarak)
        {
            transform.position = new Vector3(transform.position.x + jarak2, transform.position.y, transform.position.z);
        }
    }
}
```

Pada kode ini jarak pemindahan pada background ada yang berbeda dan ada yang sama

### 4.2.14 Game Over

Game akan berakhir ketika player mencapai titik tertentu pada permainan. Ketika game selesai akan ada panel yang menunjukkan berasa sampah yang telah dibuang serta berapa anak yang membuang sampah sembarangan. Kode berikut sebagai kode yang digunakan untuk menunjukkan panel yang akan muncul

```
public class overakhir : MonoBehaviour
{
    // Start is called before the first frame update
    Transform ambil, buang, jumlah;
    GameObject pla;
    void Start()
    {
        ambil = gameObject.transform.GetChild(5);
        buang = gameObject.transform.GetChild(6);
        jumlah = gameObject.transform.GetChild(7);
        pla = GameObject.Find("orang");
    }

    // Update is called once per frame
    void Update()
    {
        ambil.GetComponent<Text>().text = "x" + pla.GetComponent<playermove>().sampah_ambil;
        buang.GetComponent<Text>().text = "x" + pla.GetComponent<playermove>().sampah_buang;
        jumlah.GetComponent<Text>().text = pla.GetComponent<playermove>().score.ToString();
    }

    public void menu()
    {
        SceneManager.LoadScene(0);
    }
}
```

### 4.2.15 Stage 2

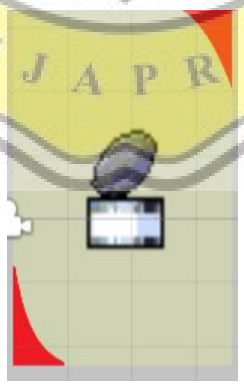
Stage 2 dibuat menjadi seperti kuis pemain akan memilih sampah dan akan di buat menjadi kuis. Pemain akan memilih sampah yang telah dipilih tadi. Jika sampah dan tempat sampah yang tersedia berbeda maka pemain akan berteriak. Jika sampah dan tempat sampahnya sama maka pemain akan diam hingga timer di bagian atas selesai.

### 4.2.16 Sampah

Akan ada 5 jenis sampah yang ditampilkan di dalam stage ini yaitu sampah kaca, plastik, kertas, logam, dan sampah b3. Setiap sampah akan mempunyai tempat sampahnya sendiri. Tempat sampah akan memiliki nama di bagian depan sebagai tanda tempat sampah itu menampung sampah apa.

### 4.2.17 Holder Gambar

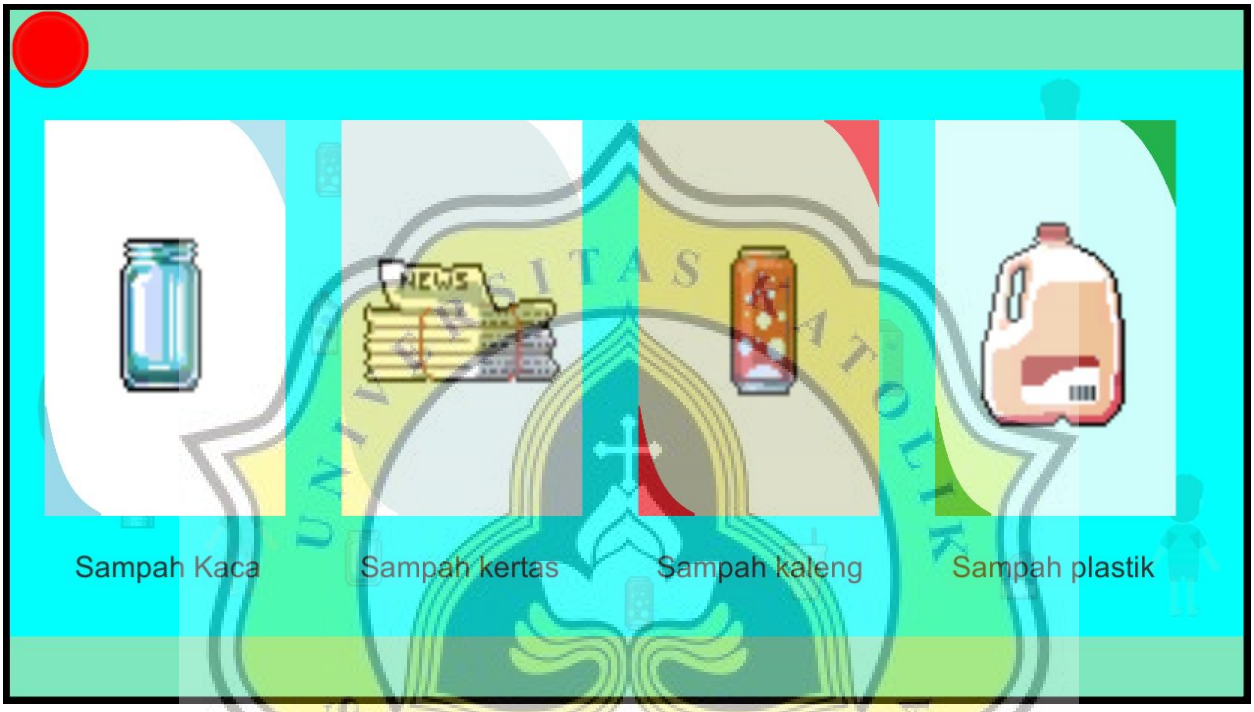
Pada stage ini akan ada tempat untuk sampah tampilan dari holder ini akan ditampilkan pada gambar 4.11



*4.11 Tampilan sampah dan holder sampah*

### 4.2.18 Memilih sampah

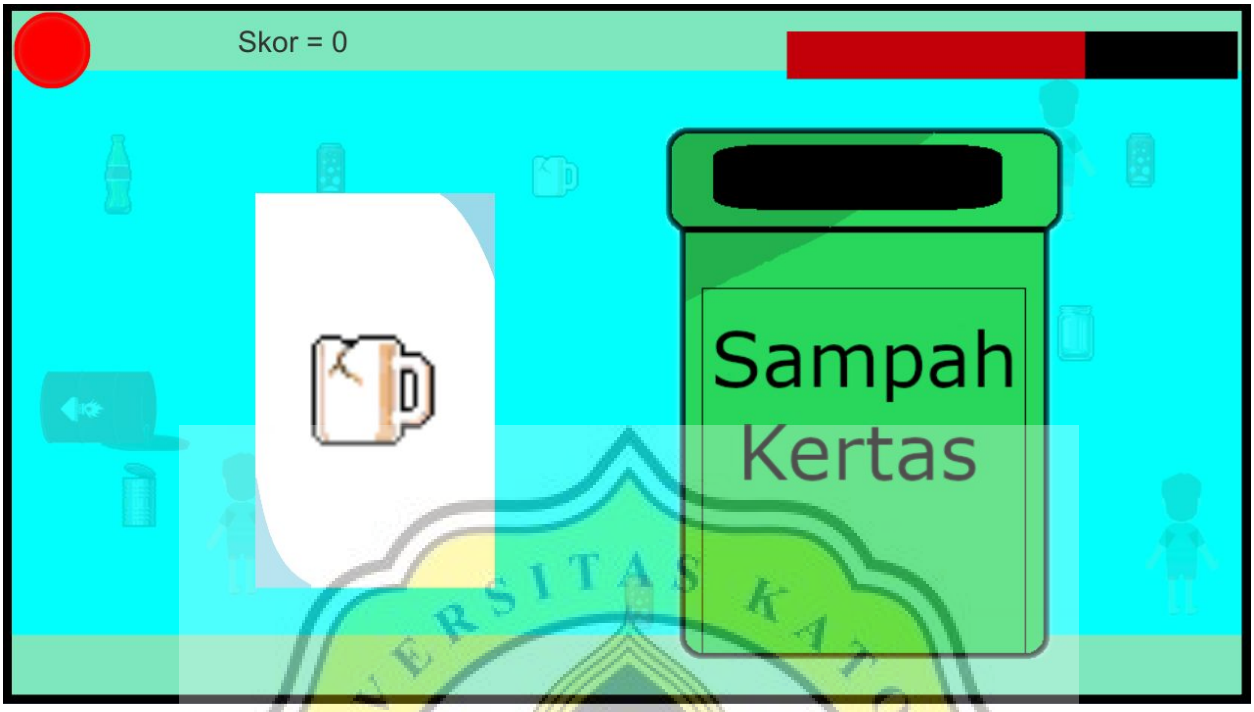
Sebelum memulai permainan pemain akan diminta untuk memilih sampah yang akan dipilah. Akan ada 4 pilihan jenis sampah dari 5 jenis yang tersedia. Gambar 4.12 akan menunjukkan tampilan ketika akan memilih sampah




*Gambar 4.12 tampilan memilih sampah*

### 4.2.19 sampah yang muncul dan tempat sampahnya

Setelah memilih sampah maka permainan akan dimulai dalam permainan ini pemain akan di tampilkan jenis sampah dan jenis tempat sampahnya. Jenis sampahnya berada pada bagian kiri dan jenis tempat sampahnya ada di sebelah kanan. Gambar 4.13 akan menunjukkan tampilan saat permainan sedang berlangsung




*Gambar 4.13 tampilan bermain*

### 4.2.20 Waktu

Waktu digunakan sebagai aturan di dalam stage 2 ini. ada 2 aturan waktu di dalam kuis aturan pertama untuk setiap pernyataan yang akan berganti di setiap jawabannya. Aturan yang kedua untuk keseluruhan stage 2. Jika aturan waktu 1 habis maka akan berganti ke pernyataan berikutnya jika aturan waktu 2 habis maka game akan berakhir. Kode berikut akan menunjukkan kode yang digunakan

```
if (!reload&&checkplay==1)
      {
         sliderx.SetActive(true);
         waktu2 -= Time.deltaTime;
         slider.gameObject.transform.localScale = new Vector3(nslidex, slider.transform.localScale.y);
         nslidex = slidex * ( waktu2 / 4) ;
         skor.SetActive(true);
         if (waktujalan == 1)
         { waktumain -= Time.deltaTime; }
      }
      sound = this.transform.GetComponent<sound>().loud;
      if (waktumain > 0)
```

```
        {
            if (reload)
            {
                waktu -= Time.deltaTime;
                if (waktu < 0)
                {
                    waktujalan = 1;
                    hapus();
                    baru();
                    reload = !reload;
                    waktu2 = 4;
                    nslidex = slidex;
                    bs.SetActive(false);
                }
}
else if (sound >= 20 && !reload)
            {
                waktujalan = 0;
                if (benar != checkholder)
                {
                    bs.SetActive(true);
                    bs.gameObject.GetComponent<SpriteRenderer>().sprite                                    =
bs.gameObject.GetComponent<benars>().bs[0];
                    soi.text = "benar";
                    reload = !reload;
                    waktu = 2;
                    score += 1;
                    bnr += 1;
                }
                else if (benar == checkholder)
                {
                    bs.SetActive(true);
                    bs.gameObject.GetComponent<SpriteRenderer>().sprite                                    =
bs.gameObject.GetComponent<benars>().bs[1];
                    soi.text = "salah";
                    reload = !reload;
                    waktu = 2;
                    slh += 1;
                }

            }

            else if (waktu2 < 0 && !reload)
            {
                waktujalan = 0;
                if (benar != checkholder)
                {
                    bs.SetActive(true);
                    bs.gameObject.GetComponent<SpriteRenderer>().sprite                                    =
bs.gameObject.GetComponent<benars>().bs[1];
                    soi.text = "salah";
                    reload = !reload;
                    waktu = 2;
```

```
                slh += 1;
            }
            else if (benar == checkholder)
            {
                bs.SetActive(true);
                bs.gameObject.GetComponent<SpriteRenderer>().sprite =
bs.gameObject.GetComponent<benars>().bs[0];
                soi.text = "benar";
                reload = !reload;
                waktu = 2;
                score += 1;
                bnr += 1;
            }
        }
    }
    else if (waktumain <= 0)
    {
        over.SetActive(true);
        checkplay = 0;
    }
}
```

### 4.2.21 Score

Pada stage 2 diberikan skor pada pemain. Skor akan bertambah jika pemain menjawab dengan benar. Salah menjawab maka skor tidak akan bertambah. Kode yang digunakan dalam membuat skor ini menyatu dengan kode yang diberikan pada waktu.

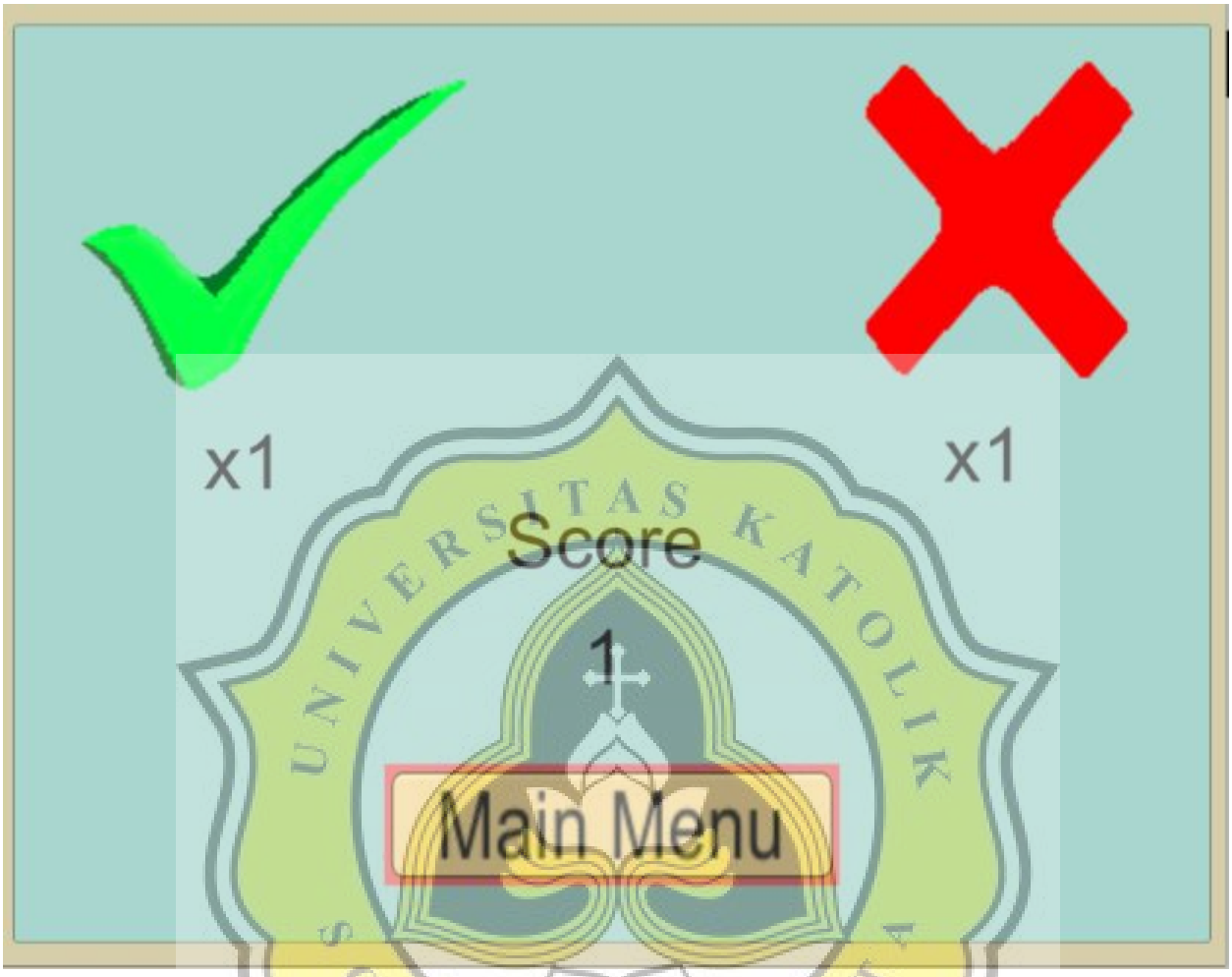
### 4.2.22 Suara (Penentu Benar salah)

Pada stage 2 tidak diberikan tombol ketika bermain. Permainan dilakukan dengan suara jika pernyataan yang diberikan salah maka pemain harus mengeluarkan suara sebelum waktu habis. Jika waktu habis pemain juga belum tentu salah karena jika pernyataan yang diberikan kepada pemain benar maka pemain tidak perlu mengeluarkan suara. Pada stage ini jika pernyataan salah maka pemain mengeluarkan suara dan jika pernyataan benar maka pemain tidak mengeluarkan suara. Akan ada tanda jika pemain benar atau salah di setiap pernyataan. Gambar 4.14 akan menunjukkan tampilan jika jawaban salah


*Gambar 4.14 tampilan jika jawaban salah*

### 4.2.23 Game Over

Jika waktu untuk keseluruhan game yang diberikan habis maka permainan akan selesai.setelah waktu habis maka akan diberikan panel yang memberitahu skor dan jumlah jawaban benar dan salah. Panel juga memberikan tombol main menu untuk kembali ke main menu. Gambar 4.15 akan menunjukkan tampilan panel yang ditunjukkan ketika permainan selesai




*Gambar 4.15 tampilan panel*

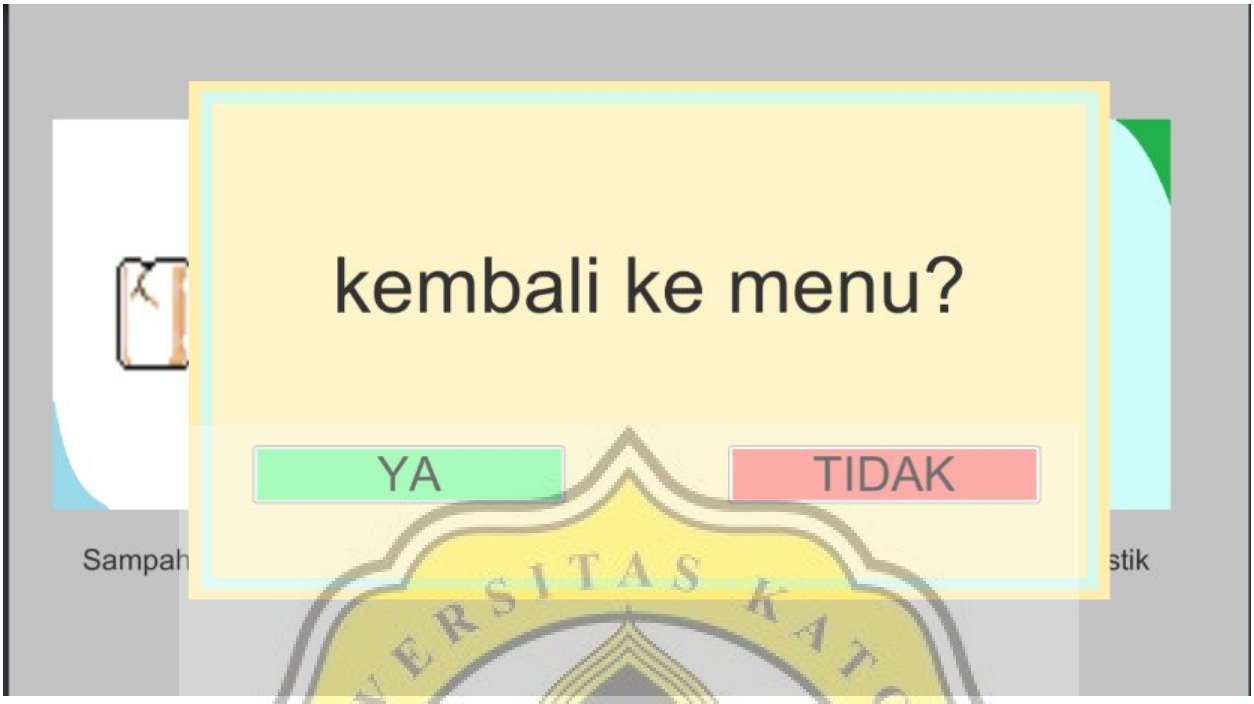
### 4.2.24 Cara Main

Pada bagian cara main akan menjelaskan stage 1 dan stage 2. Gambar 4.16 akan memperlihatkan tampilan dari salah satu tampilan dari cara bermain.

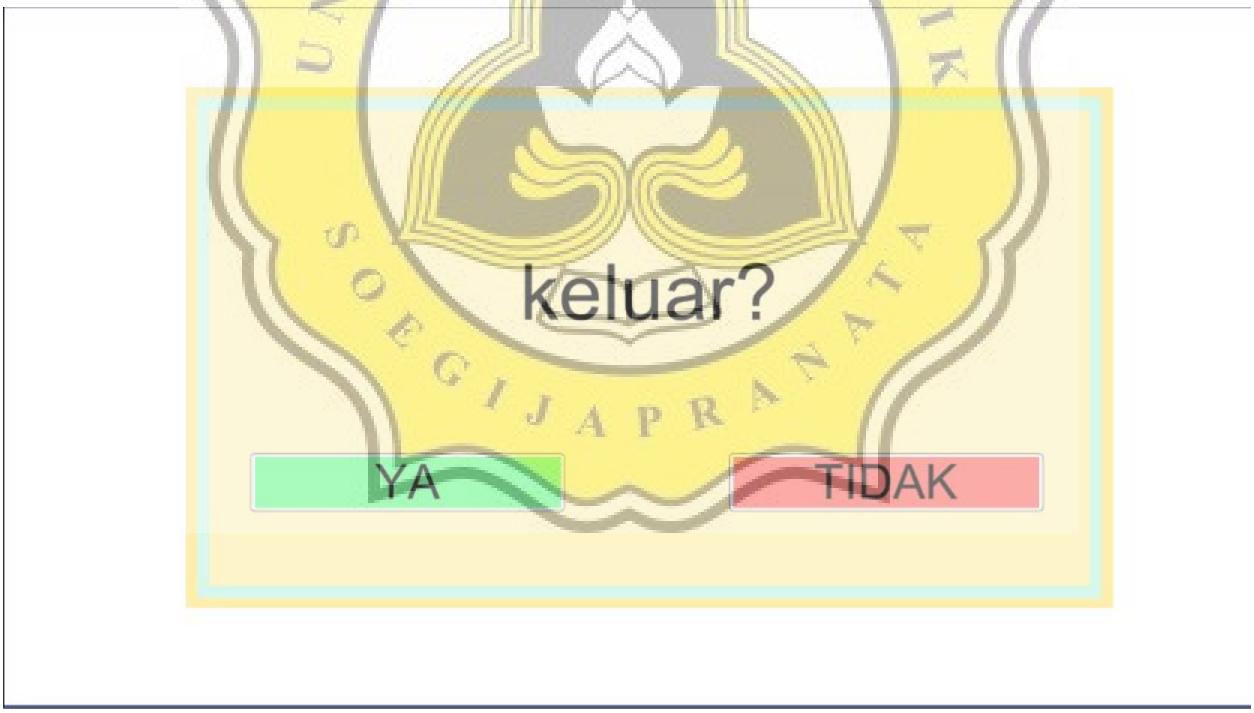

*4.16 Cara bermain*

### 4.2.25 Exit (Game atau Level)

Pemain dapat kembali ke main menu jika berada di dalam game dan jika berada di main menu maka player dapat keluar dari game. Jika ingin keluar maka player dapat menekan tombol back pada android. Gambar 4.16 dan 4.17 akan menunjukkan tampilan ketika ingin kembali ke main menu dan ketika akan keluar game.




*Gambar 4.17 Tampilan kembali ke main main*




*Gambar 4.18 tampilan keluar dari game*

UNIVERSITAS KATOLIK SOEGIJAPRANATA

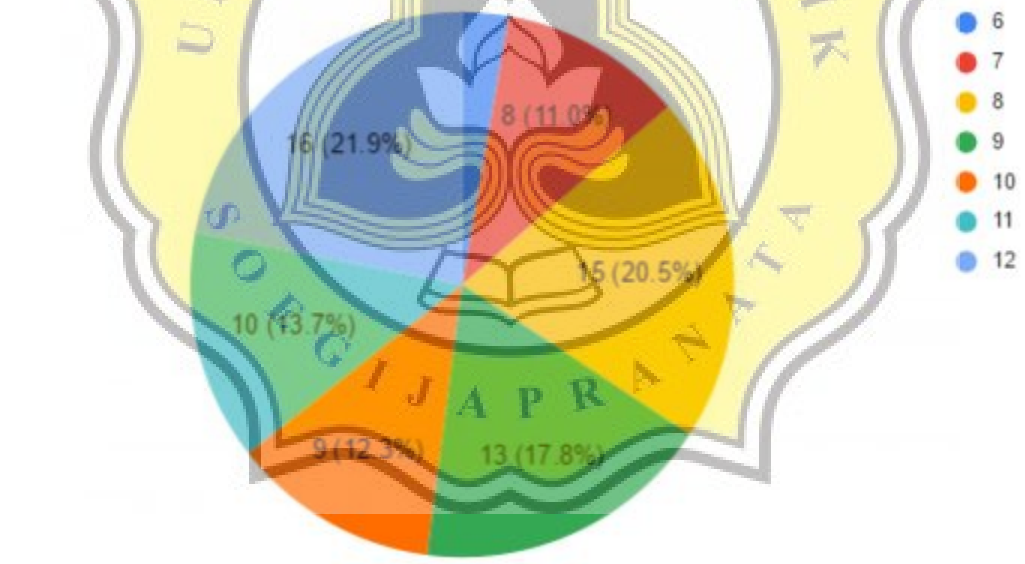
### 4.3 Analisa dan Responden

Setelah *game* selesai dan tidak ada lagi masalah dalam gamenya, dilakukan percobaan *game* dan melakukan pengumpulan data dengan menyebarkan kuesioner ke 73 orang yang masih berada di tahap pendidikan sekolah dasar. Didapatkan hasil sebagai berikut:

#### 4.3.1 Profil Responden

1. Umur

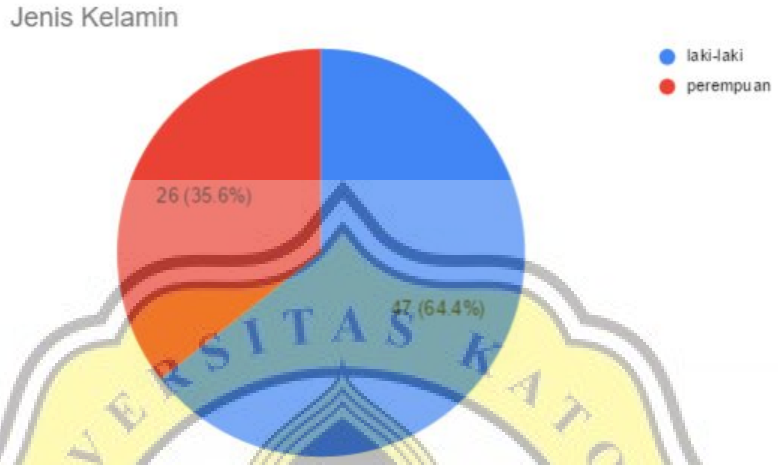
   Responden berasal dari umur 6 sampai umur 12 tahun gambar 4.19 menunjukkan diagram umur dari responden.




*Gambar 4.19 diagram pie umur*

2. Jenis Kelamin

Responden ada laki dan perempuan berasal dari 73 responden berasal dari 26 perempuan dan 47 laki-laki gambar 4.20 akan menunjukkan diagram jenis kelamin dari responden

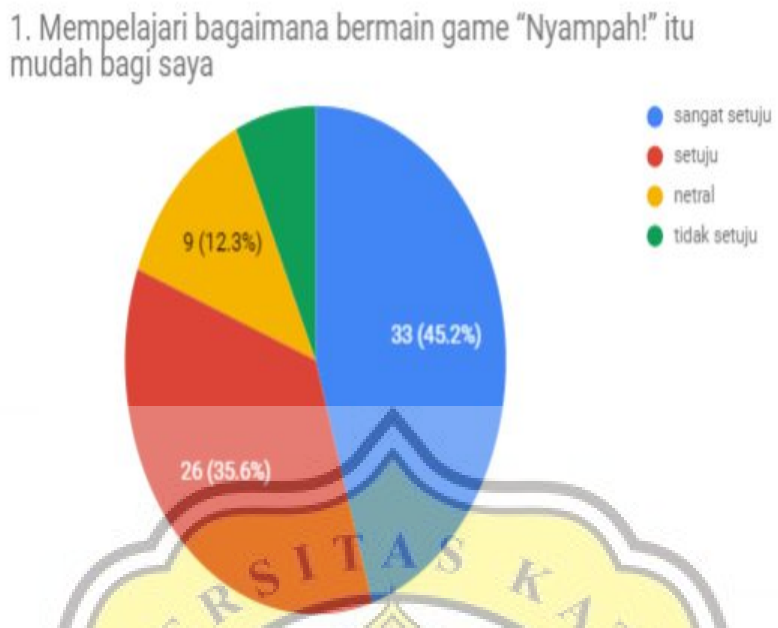



*Gambar 4.20 diagram jenis kelamin*

### 4.3.2 Pembahasan Data kuesioner Terhadap Game

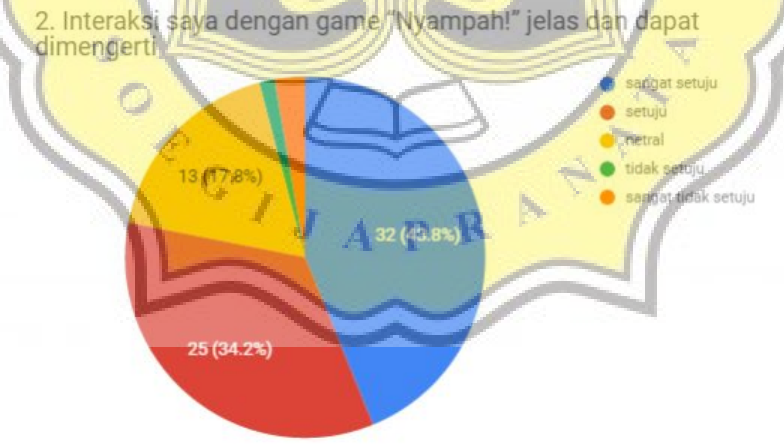
Data kuesioner yang didapatkan dari permainan game "nyampah!" Dapat dilihat dibawah ini:

Gambar 4.21 memperlihatkan 0% sangat tidak setuju, 6.8% tidak setuju, 12,3,% netral, 35.6 setuju, dan 45,2% sangat setuju dengan pernyataan mempelajari bagaimana bermain game "Nyampah!" itu mudah bagi saya.



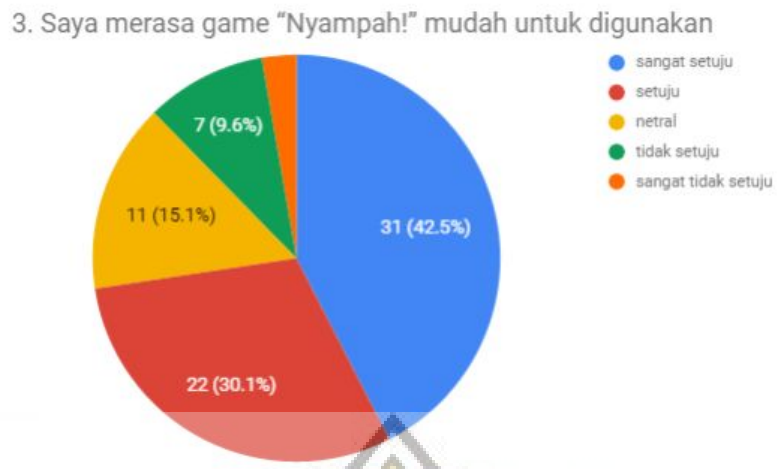

*Gambar 4.21 Diagram EE1*

Gambar 4.22 memperlihatkan 2,7% sangat tidak setuju, 1,4% tidak setuju, 17,8% netral, 34.2% setuju, dan 43.8% sangat setuju dengan pernyataan Interaksi saya dengan game "Nyampah!" jelas dan dapat dimengerti.




*Gambar 4.22 Diagram EE2*

Gambar 4.23 memperlihatkan 2,7% sangat tidak setuju,9,6% tidak setuju, 15,1% netral, 30,1% setuju, dan 42,5% sangat setuju dengan pernyataan saya merasa game "Nyampah!" mudah untuk digunakan.

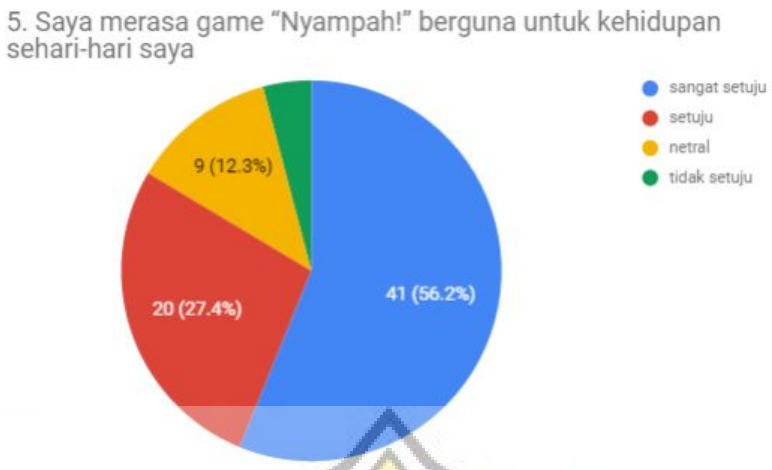



*Gambar 4.23 Diagram EE3*

Gambar 4.24 memperlihatkan 5,5% sangat tidak setuju,13.7 tidak setuju, 5,5% netral, 31,5% setuju, dan 43,8% sangat setuju dengan pernyataan mudah bagi saya untuk mahir memainkan game "Nyampah!".


*Gambar 4.24 Diagram EE4*

Gambar 4.25 memperlihatkan 0% sangat tidak setuju, 4,1% tidak setuju, 12,3% netral, 27,4% setuju, dan 56,2% sangat setuju dengan pernyataan saya merasa game "Nyampah!" berguna untuk kehidupan sehari-hari saya..

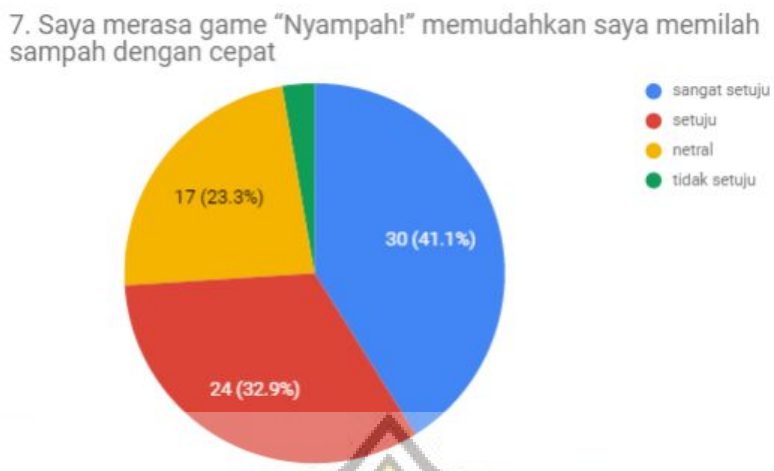



*Gambar 4.25 Diagram PE1*

Gambar 4.26 memperlihatkan 1,4% sangat tidak setuju, 0% tidak setuju, 12,3% netral, 34,2% setuju, dan 52,1% sangat setuju dengan pernyataan saya merasa game "Nyampah!"membantu saya belajar membedakan sampah..

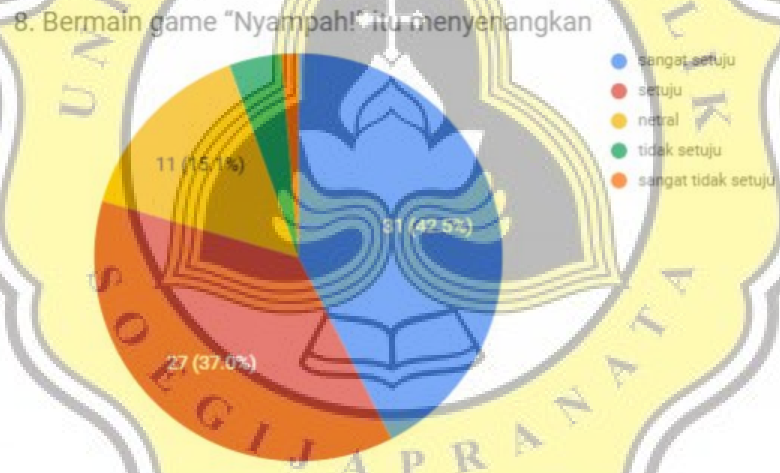

*Gambar 4.26 Diagram PE2*

Gambar 4.27 memperlihatkan 0% sangat tidak setuju, 2,7% tidak setuju, 23,3% netral, 32,9% setuju, dan 41,1% sangat setuju dengan pernyataan saya merasa game "Nyampah!" memudahkan saya memilah sampah dengan cepat..



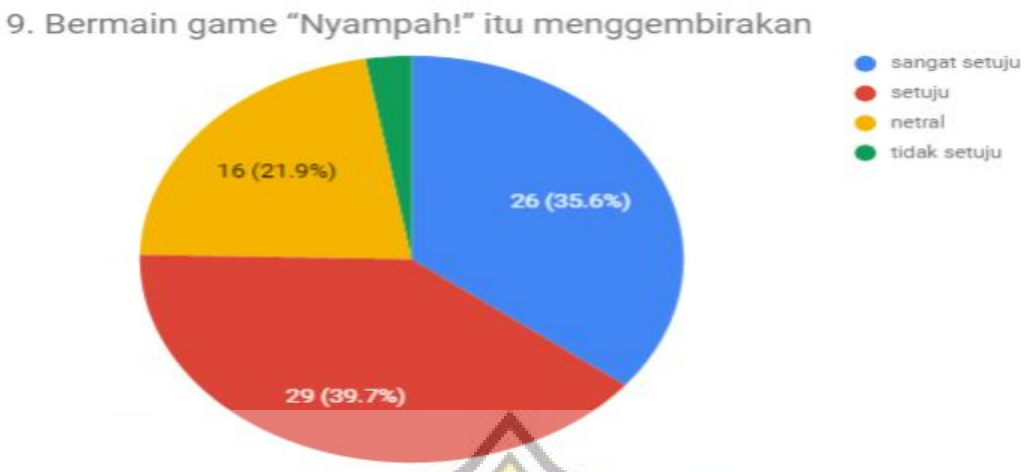

*Gambar 4.27 Diagram PE3*

Gambar 4.28 memperlihatkan 1,4% tidak setuju, 4,1% tidak setuju, 15,1% netral, 37% setuju, dan 42,5% sangat setuju dengan pernyataan bermain game "Nyampah!" itu menyenangkan.



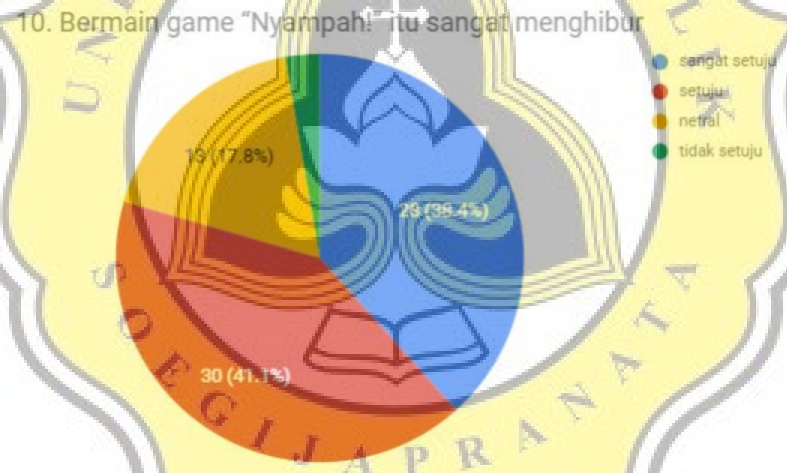

*Gambar 4.28 Diagram HM1*

Gambar 4.29 memperlihatkan 0% sangat tidak setuju, 2,7% tidak setuju, 21,9% netral, 39,7% setuju, dan 35,6% sangat setuju dengan pernyataan bermain game "Nyampah!" itu menggembirakan.




*Gambar 4.29 Diagram HM2*

Gambar 4.30 memperlihatkan 0% sangat tidak setuju, 2,7% tidak setuju , 17,8% netral, 41,1% setuju, dan 38,4% sangat setuju dengan pernyataan bermain game "Nyampah!" itu menghibur.

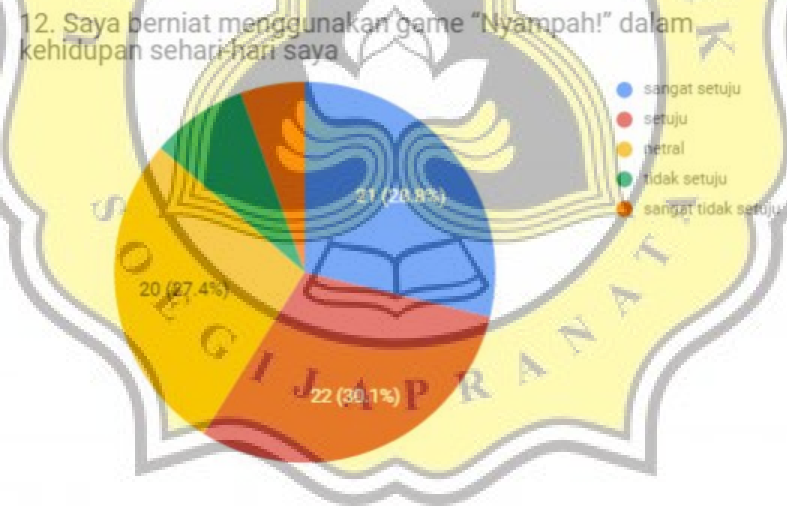



*Gambar 4.30 Diagram HM3*

Gambar 4.31 memperlihatkan 6.8% sangat tidak setuju, 15,1% tidak setuju, 41,1% netral, 17,8% setuju, dan 19,2% sangat setuju dengan pernyataan saya berencana untuk terus menggunakan game "Nyampah!" sesering mungkin di masa yang akan datang.

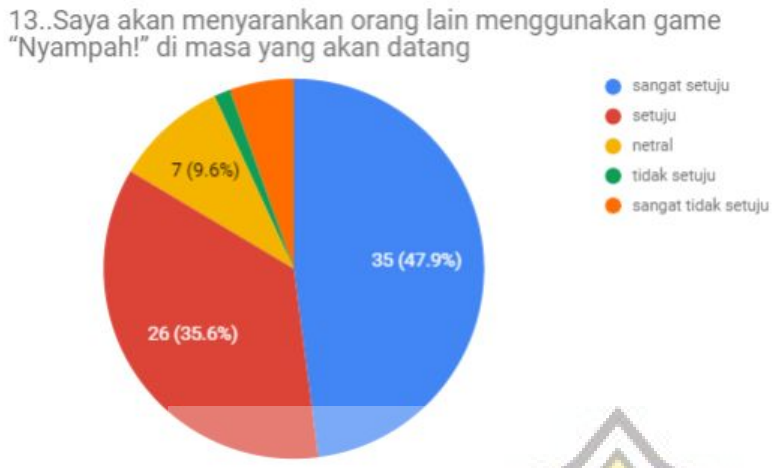

*Gambar 4.31 Diagram BI1*

Gambar 4.32 memperlihatkan 5,5% sangat tidak setuju, 8,2% tidak setuju, 27,4% netral, 30.1% setuju, dan 28,8% sangat setuju dengan pernyataan saya berniat menggunakan game "Nyampah!" dalam kehidupan sehari-hari saya.




*Gambar 4.32 Diagram BI2*

Gambar 4.33 memperlihatkan 5,5% sangat tidak setuju, 1,4% tidak setuju, 9,6% netral, 35.6% setuju, dan 47,9% sangat setuju dengan pernyataan saya akan menyarankan orang lain menggunakan game "Nyampah!" di masa yang akan datang.




*Gambar 4.33 Diagram BI3*

### 4.3.3 Model Variabel

Gambar 4.34 berikut merupakan gambar model variabel penelitian yang variabel independennya adalah EE, PE, dan HM dan variabel dependen nya adalah BI


*Gambar 4.34 Model variabel*

### 4.3.4 Uji Validitas

Gambar 4.35 menunjukkan tabel uji validitas dapat dilihat dari tabel tersebut variabel sudah berkelompok dan nilai variabel sudah di atas 0,7. Variabel yang valid mempunyai angka diatas 0,7 dan akan ditandai dengan warna hijau dan variabel yang tidak valid diberi warna merah. Pada variabel BI semua variabel sudah berada di angka 0,7 - 0,9 dan diberi warna hijau, pada variabel EE sudah berada di angka 0,81 - 0,87 dan diberi warna hijau, pada variabel HM sudah berada di angka 0,8 dan 0,9 dan diberi warna hijau, dan pada variabel PE sudah berada di angka 0,7 - 0,8 dan diberi warna hijau.

| | BI | EE | HM | PE |
|---|---|---|---|---|
| BI1 | 0.7200 | | | |
| BI2 | 0.9061 | | | |
| BI3 | 0.7800 | | | |
| EE1 | | 0.8168 | | |
| EE2 | | 0.8512 | | |
| EE3 | | 0.8250 | | |
| EE4 | | 0.8792 | | |
| HM1 | | | 0.9126 | |
| HM2 | | | 0.8866 | |
| HM3 | | | 0.8731 | |
| PE1 | | | | 0.7782 |
| PE2 | | | | 0.7880 |
| PE3 | | | | 0.8797 |

*Gambar 4.35 tabel uji validitas*

### 4.3.5 Uji Reliabilitas

dapat dilihat variabel EE, PE,HM dan BI memiliki nilai yang dapat diterima. Variabel EE memiliki nilai 0,8648. Variabel PE memiliki nilai 0,7539. Variabel HM memiliki nilai 0,8734. Dan variabel BI memiliki nilai 0,7257. Variabel dinyatakan reliabel jika berada nilai variabel berada di angka 0,7

| | Cronbach's Alpha | rho_a | Reliabel |
|---|---|---|---|
| BI | 0.7257 | 0.7485 | Reliabel |
| EE | 0.8648 | 0.8736 | Reliabel |
| HM | 0.8734 | 0.9241 | Reliabel |
| PE | 0.7539 | 0.7952 | Reliabel |

*Gambar 4.36 tabel uji reliabilitas variabel EE PE HM BI (dari atas ke bawah)*

### 4.3.6 Uji Hipotesis

Gambar 4.27 memperlihatkan P Values dari variabel EE PE dan HM dengan variabel BI. pada tabel dibawah dapat dilihat bahwa P Values EE , PE. HM, dengan BI berada di bawah angka 0,05 yang mengatakan bahwa Variabel EE,PE,dan HM memiliki pengaruh dengan Variabel BI

| | Original Sa... | Sample Me... | Standard D... | T Statistics (... | P Values |
|---|---|---|---|---|---|
| EE -> BI | 0.6493 | 0.6503 | 0.0729 | 8.9068 | 0.0000 |
| HM -> BI | -0.1418 | -0.1392 | 0.0550 | 2.5772 | 0.0102 |
| PE -> BI | 0.3753 | 0.3790 | 0.0705 | 5.3272 | 0.0000 |

*Gambar 4.37 tabel hasil uji korelasi*