BAB 4

PEMBAHASAN

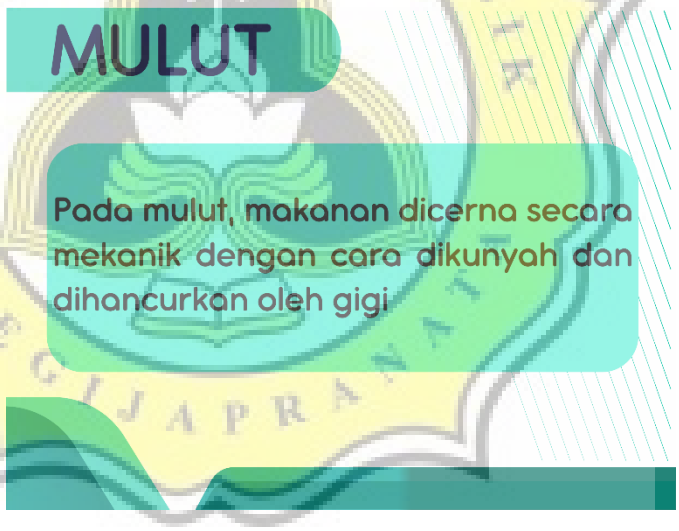
4.1 Perumusan Gameplay Game "Mari Mencerna"

"Mari Mencerna" merupakan *game* yang dibuat untuk mempermudah pemain dalam memahami materi mata pelajaran IPA berkaitan dengan sistem pencernaan manusia dan membuat pemain lebih antusias untuk belajar. Game "Mari Mencerna" menggunakan tampilan visual 2D dan berbasis Kinect untuk pengoperasian pada *game*.

Pada *game* ini tiap pemain dituntut untuk dapat menyelesaikan tantangan-tantangan yang ada pada tiap *stage*. Pada game ini terdapat 6 *stage* yang merepresentasikan organ-organ yang dipakai dalam sistem pencernaan pada manusia. Tiap stage diawali dengan pengenalan singkat mengenai stage tersebut, selanjutnya terdapat instruksi cara menyelesaikan *stage* tersebut sekaligus penjelasan singkat yang berkaitan dengan materi sistem pencernaan. Jika pemain telah berhasil menyelesaikan tantangan pada *stage* sebelum waktu habis, maka pemain akan dibawa pada *scene* informasi yang berisi pengetahuan mengenai organ sistem pencernaan manusia pada *stage* tersebut. Ketika pemain berhasil maka akan mendapatkan score, score semakin besar bila pemain semakin cepat menyelesaikan tantangan pada stage tersebut. Jika pemain gagal akan masuk ke *scene game over* Berikut penjelasan *gameplay* dari tiap *stage*:

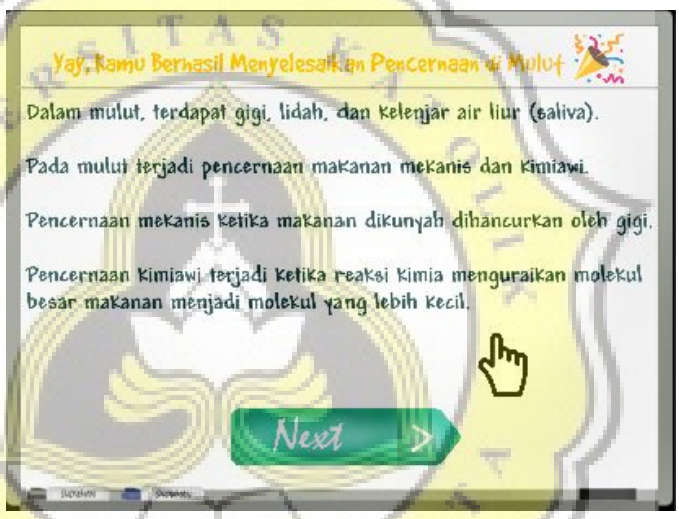
a. *Stage* 1

Stage ini merupakan bagian awal pada tahapan sistem pencernaan, organ yang direpresentasikan adalah mulut. Pada awal stage akan muncul tampilan seperti pada gambar 4.1. Pemain harus menggerakan rahang atas dan rahang bawah untuk dapat mengunyah makanan. Untuk menggerakan rahang atas dan rahang bawah pemain harus menepukkan kedua tangannya dengan posisi tangan kanan lebih tinggi dari tangan kiri. Tampilan *gameplay* dapat dilihat pada gambar 4.2. Setelah 5 kali mengunyah maka pemain berhasil menyelesaikan tantangan dan akan muncul *scene* informasi yang menjelaskan mengenai pencernaan pada organ mulut seperti pada gambar 4.3.



Gambar 4.1 Tampilan Awal Stage 1

Gambar 4.2 Gameplay Stage 1

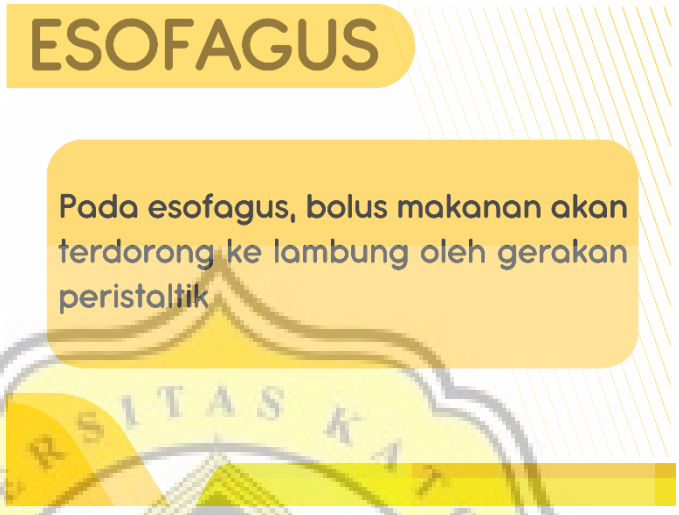


Gambar 4.3 Scene Informasi Stage 1

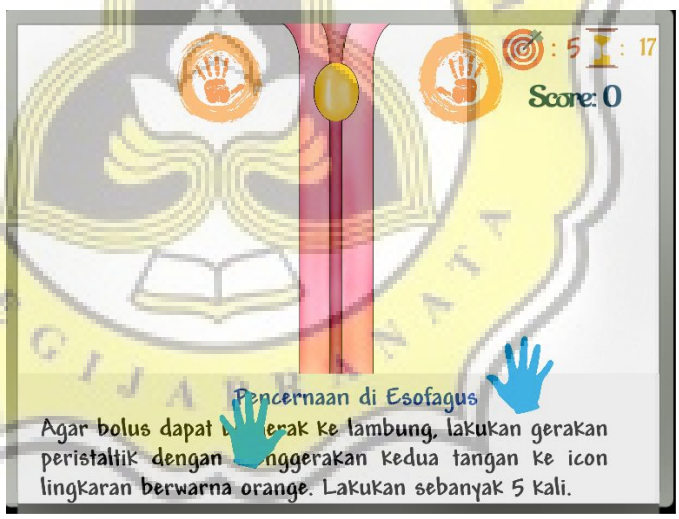
b. *Stage* 2

Pada *stage* 2 ini organ yang direpresentasikan adalah esofagus (kerongkongan). Pada awal stage akan muncul tampilan seperti pada gambar 4.4 sebagai pengantar stage 2 ini. Pemain harus mengarahkan kedua tangan untuk menyentuh *icon* lingkaran jingga untuk merepresentasikan gerakan peristaltik. *Gameplay* dapat dilihat pada gambar 4.5. Setelah pemain berhasil melakukannya bolus akan turun, setelah 5 kali melakukan gerakan peristaltik pemain berhasil

menyelesaikan tantangan dan akan muncul *scene* informasi yang menjelaskan mengenai pencernaan pada organ esofagus seperti pada gambar 4.6.



Gambar 4.4 Tampilan Awal Stage 2

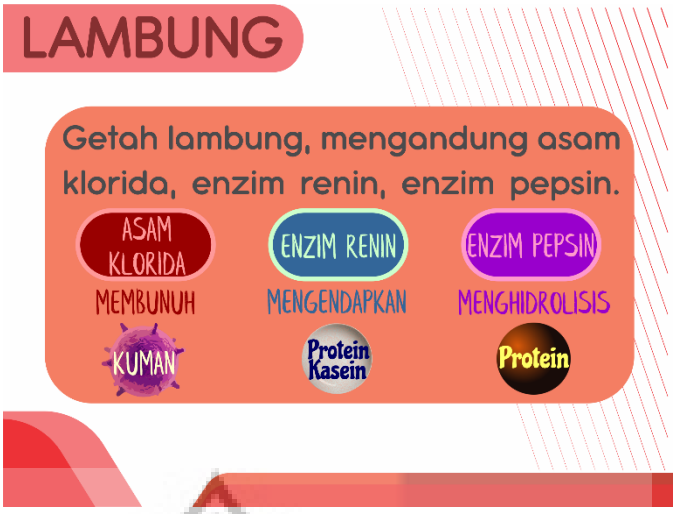


Gambar 4.5 Gameplay Stage 2

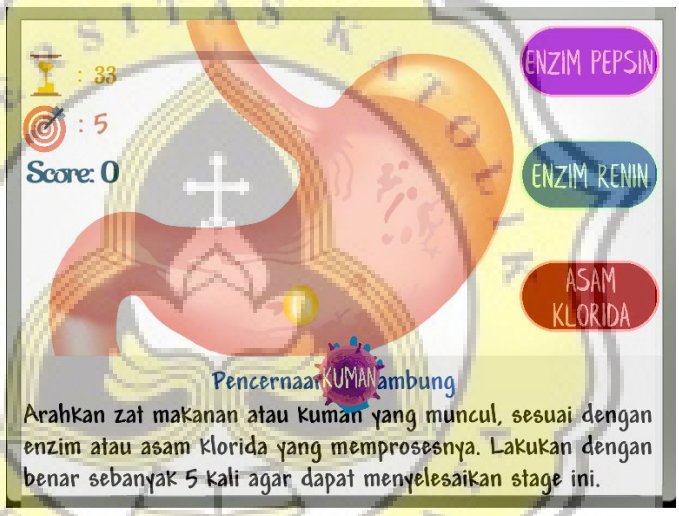
Gambar 4.6 Scene Informasi Stage 2

c. *Stage* 3

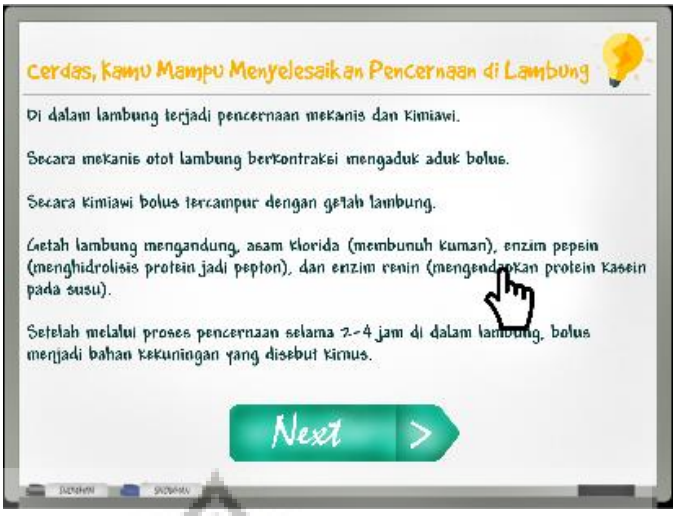
Pada *stage* 3 ini organ yang direpresentasikan adalah lambung. Pada awal stage akan muncul tampilan seperti gambar 4.7. Tampilan *gameplay* untuk *stage* 3 dapat dilihat pada gambar 4.8. Pemain harus mengarahkan zat ataupun kuman yang keluar dari makanan ke kapsul enzim maupun asam klorida yang memproses zat makanan tersebut. Bila salah maka pemain harus mencoba lagi untuk mengarahkan zat makanan dalam kapsul. Bila benar maka target yang harus dicapai akan berkurang. Bila target telah 0, maka akan muncul *scene* informasi yang menjelaskan mengenai pencernaan pada organ lambung seperti pada gambar 4.9.



Gambar 4.7 Tampilan Awal Stage 3



Gambar 4.8 Gameplay Stage 3

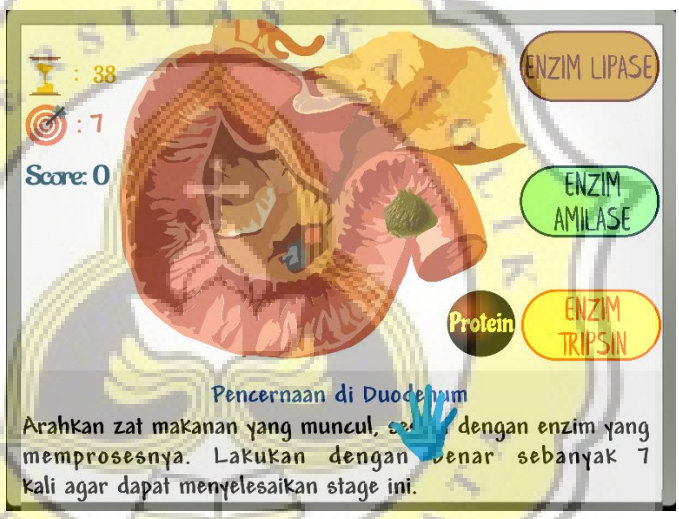


Gambar 4.9 Scene Informasi Stage 3

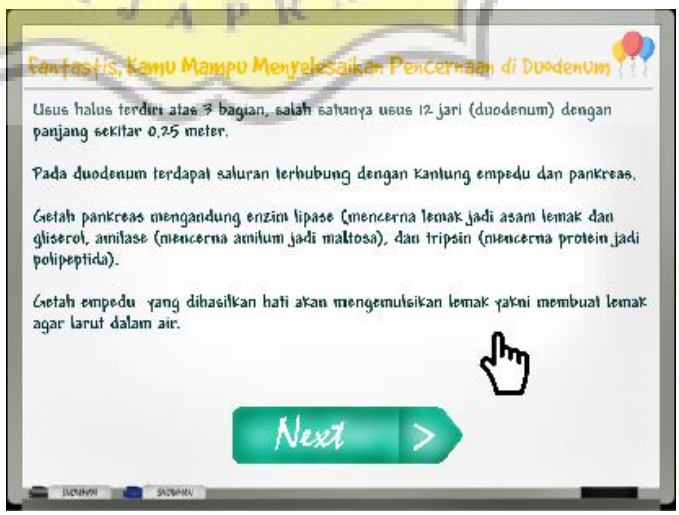
d. *Stage* 4

Pada *stage* 4 ini organ yang direpresentasikan adalah duodenum (usus 12 jari). Tampilan awal stage seperti pada gambar 4.10. Pada bagian ini pemain harus mengarahkan zat yang keluar dari kimus ke dalam kapsul-kapsul yang melambangkan enzim dan asam klorida. Arahkan zat sesuai dengan enzim atau asam klorida yang akan memprosesnya. *Gameplay* stage 4 dapat dilihat pada gambar 4.11. Bila salah maka pemain harus mencoba lagi untuk mengarahkan zat makanan dalam kapsul. Bila benar maka target yang harus dicapai akan berkurang. Bila target telah 0, maka akan muncul *scene* informasi yang menjelaskan mengenai pencernaan pada organ duodenum seperti pada gambar 4.12.

Gambar 4.10 Tampilan Awal Stage 4



Gambar 4.11 Gameplay Stage 4



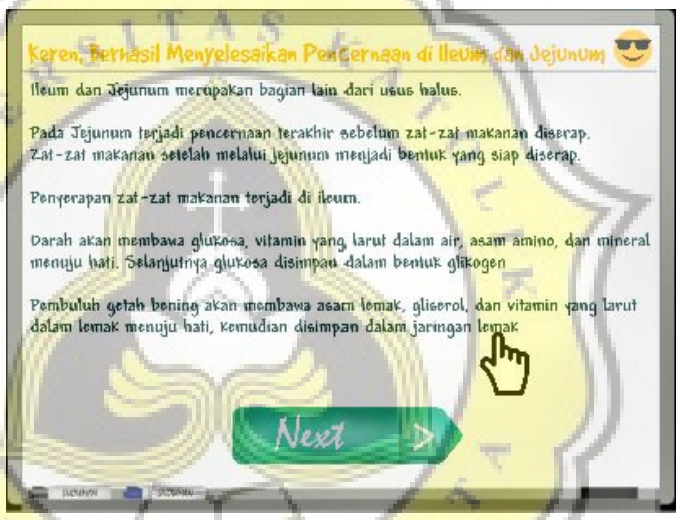
Gambar 4.12 Scene Informasi Stage 4

e. *Stage* 5

Pada *stage* 5 ini organ yang direpresentasikan adalah ileum dan jejunum, sesuai judul tampilan awal dapat dilihat pada gambar 4.13. Pada bagian ini pemain harus mengarahkan zat yang keluar dari makanan ke kapsul enzim yang memproses zat makanan tersebut seperti pada gambar 4.14. Bila salah maka pemain harus mencoba lagi untuk mengarahkan zat makanan dalam kapsul. Bila benar maka target yang harus dicapai akan berkurang. Bila target telah 0, maka akan muncul *scene* informasi yang menjelaskan mengenai pencernaan pada organ usus halus (ileum dan jejunum). Tampilan *scene* informasi dapat dilihat pada gambar 4.15.

Gambar 4.13 Tampilan Awal Stage 5

Gambar 4.14 Gameplay Stage 5

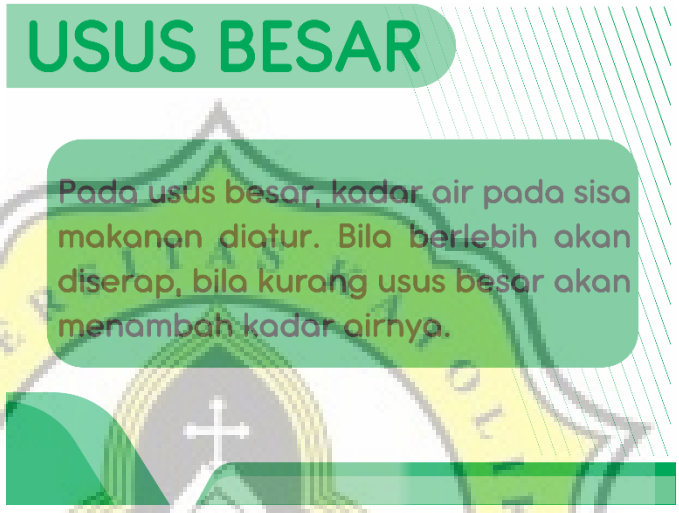


Gambar 4.15 Scene Informasi Stage 5

f. *Stage* 6

Pada *stage* 6 ini, organ yang direpresentasikan adalah usus besar. Tampilan awal dari stage 6 dapat dilihat pada gambar 4.16. Pemain diharuskan mengatur kadar air pada kotoran sehingga mencapai indikator normal. Bila air terlalu banyak maka kotoran akan berubah menjadi terlalu cair, sedangkan bila air terlalu sedikit maka kotoran akan berubah menghitam menjadi terlalu padat. Secara berkala kadar air di kotoran akan berkurang, pemain diharuskan memencet tombol

air secara bergiliran untuk menambah kadar air pada kotoran. Tampilan *gameplay* dapat dilihat pada gambar 4.17. Bila pemain berhasil mempertahankan kadar air normal hingga mencapai rektum, maka akan muncul *scene* informasi yang menjelaskan mengenai pencernaan pada organ usus besar seperti pada gambar 4.18.

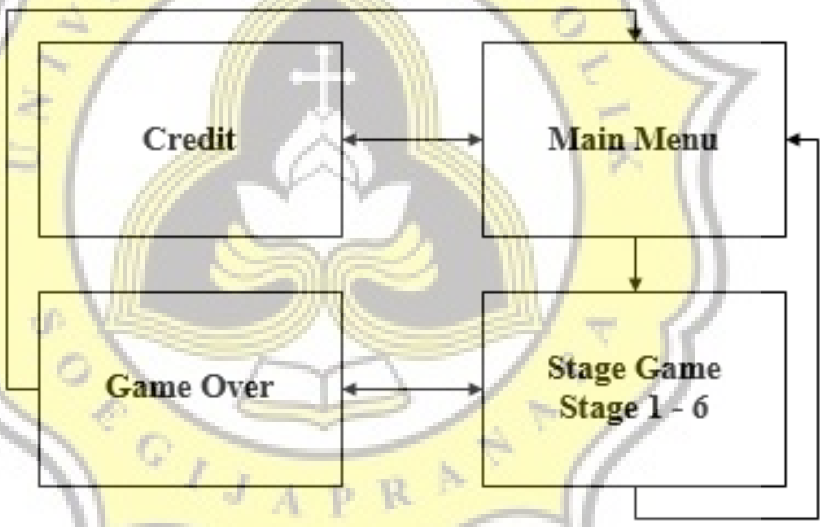


Gambar 4.16 Tampilan Awal Stage 6

Gambar 4.17 Gameplay Stage 6

Gambar 4.18 Scene Informasi Stage 6

Gambaran alur permainan dari *game* "Mari Mencerna" dapat dilihat pada gambar 4.19.



Gambar 4.19 Alur Permainan

4.2 Asset Game "Mari Mencerna"

Berikut merupakan asset yang digunakan pada *game* "Mari Mencerna":

4.2.1 Main Menu

Gambar 4.20 merupakan tampilan dari *main menu* yang terdapat pada *game* "Mari Mencerna".

Gambar 4.20 Tampilan Main Menu

Tampilan background untuk *main menu* dapat dilihat pada gambar 4.21. Dan desain tombol yang digunakan pada *main menu* dapat dilihat pada gambar 4.22. Tombol *play* berfungsi untuk memulai permainan, sedangkan tombol *help* digunakan untuk berpindah ke *help section* yang berisi petunjuk mengenai perolehan score dan cara berpindah stage secara otomatis ketika telah menyelesaikan stage tertentu, tombol *quit* digunakan untuk keluar dari *game*. Tombol info digunakan untuk masuk ke bagian credit scene.

Gambar 4.21 Background Main Menu

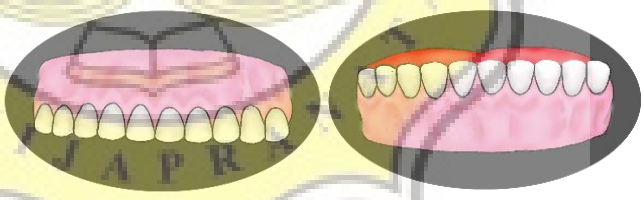
Gambar 4.22 Tombol Play, Help, Quit, dan Info

4.2.2 Stage 1

Pada *stage* 1, pencernaan dilakukan di mulut, oleh karena itu tampilan desain *gameplay* menggunakan rahang atas dan bawah seperti pada gambar 4.23.

Gambar 4.23 Rahang Atas dan Bawah

Untuk bagian awal *stage* 1, kursor yang digunakan untuk menandai tangan kanan dan kiri juga menggunakan gambar rahang atas dan bawah seperti pada gambar 4.24 ini.



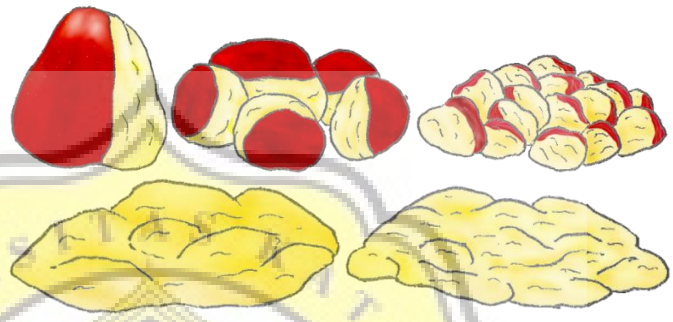
Gambar 4.24 Kursor Rahang Atas dan Bawah

Pada *stage* 1 pemain diminta untuk menggerakan rahang agar dapat mengunyah apel dengan menepukan tangan sesuai dengan instruksi pada gambar 4.25. Gambar 4.26 memperlihatkan gambaran perubahan apel saat proses mengunyah berlangsung.

Pencernaan di Mulut

Untuk mengunyah makanan tepukan kedua tangan dengan posisi tangan kanan di atas agar rahang bergerak. Lakukan sebanyak 5 kali.

Gambar 4.25 Instruksi Stage 1



Gambar 4.26 Gambaran Perubahan Apel

Setelah pemain mampu menyelesaikan tantangan pada *stage* 1, maka akan muncul tampilan informasi berkaitan dengan pencernaan di mulut seperti pada gambar 4.27.

Yay, Kamu Berhasil Menyelesaikan Pencernaan di Mulut

Dalam mulut, terdapat gigi, lidah, dan kelenjar air liur (saliva).

Pada mulut terjadi pencernaan makanan mekanis dan kimiawi.

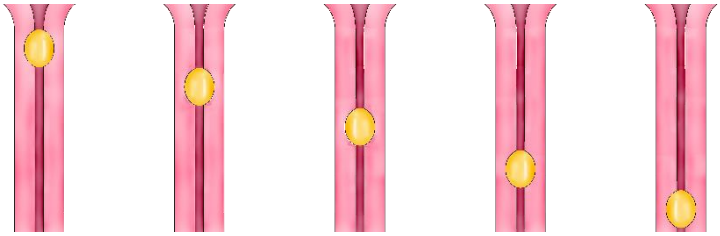
Pencernaan mekanis ketika makanan dikunyah dihancurkan oleh gigi.

Pencernaan kimiawi terjadi ketika reaksi kimia menguraikan molekul besar makanan menjadi molekul yang lebih kecil.

Gambar 4.27 Tampilan Informasi Stage 1

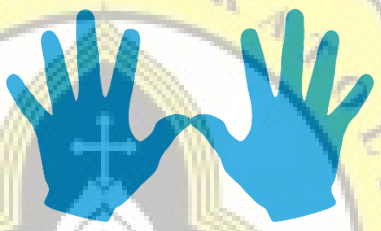
4.2.3 Stage 2

Pada *stage* 2, tahapan pencernaan dilakukan di esofagus, oleh karena itu tampilan desain *gameplay* menggunakan tampilan esofagus beserta bolus seperti pada gambar 4.28.



Gambar 4.28 Esofagus beserta bolus

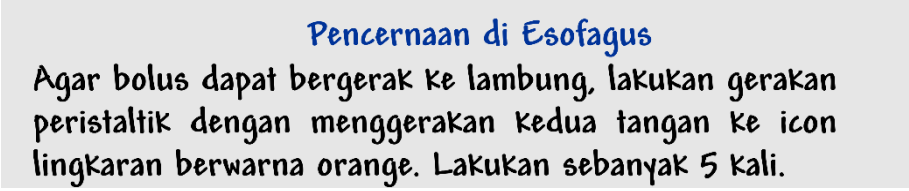
Gerakan yang harus dilakukan pada *stage* 2 menggunakan kedua tangan pemain. Gambar 4.29 menunjukkan kursor tangan kanan dan kiri yang digunakan pemain dalam permainan. Untuk *asset* kursor juga digunakan pada *stage* 3 hingga *stage* 6.



Gambar 4.29 Kursor Tangan Kanan dan Kiri

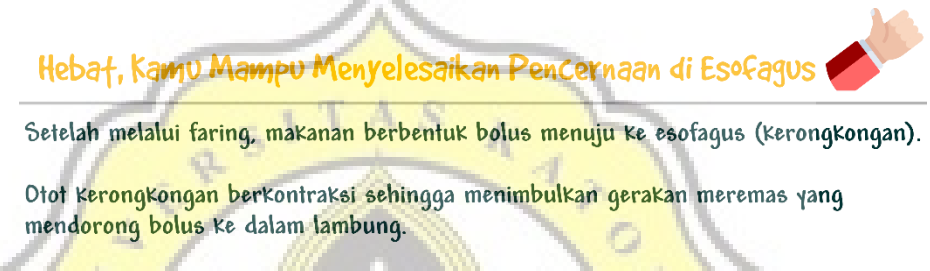
Pada *stage* 2 ini, pemain harus menggerakan kedua tangan untuk menyentuh *icon* lingkaran bergambar tangan yang berwarna jingga, seperti pada gambar 4.30. Sehingga nanti bolus akan bergerak turun kearah lambung karena mengalami gerakan peristaltik. Instruksi ini dapat ditemukan pada instruksi *game* seperti pada gambar 4.31.

Gambar 4.30 Tangan Kanan dan Kiri untuk Gerakan Peristaltik



Gambar 4.31 Instruksi Stage 2

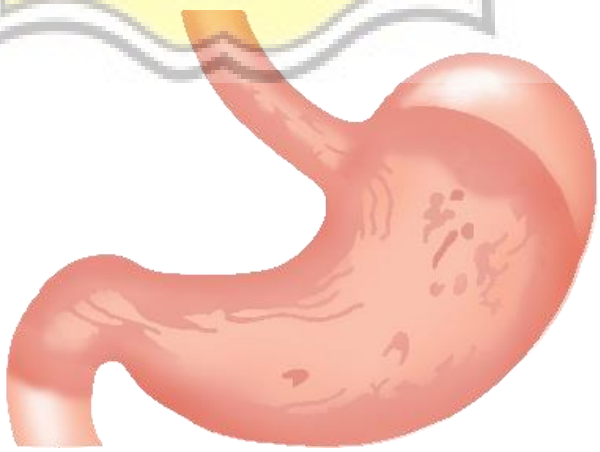
Setelah pemain telah berhasil menyelesaikan tantangan pada *stage* 2, maka akan muncul tampilan informasi berkaitan dengan pencernaan di esofagus seperti pada gambar 4.32.



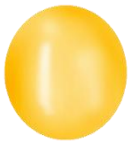
Gambar 4.32 Tampilan Informasi Stage 2

4.2.4 Stage 3

Pada *stage* 3, pencernaan dilakukan di lambung, oleh karena itu tampilan desain *gameplay* menggunakan tampilan lambung seperti pada gambar 4.33 dan disertai dengan tampilan bolus yang diperoleh dari esofagus pada gambar 4.34.

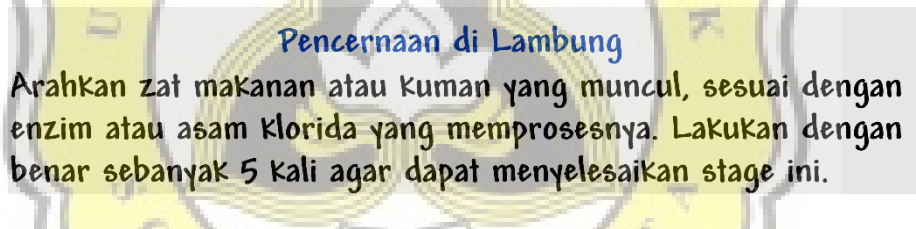


Gambar 4.33 Lambung



Gambar 4.34 Bolus

Di *stage* 3 ini, pemain harus mengarahkan zat atau kuman yang berasal dari bolus menuju ke asam klorida atau enzim yang akan memproses zat atau kuman tersebut. Instruksi tersebut dapat dilihat pada gambar 4.35. Tampilan zat makanan dan kuman yang berasal dari bolus seperti pada gambar 4.36, untuk gambar zat protein juga dipakai dalam *stage* 4. Kemudian untuk tampilan enzim dan asam klorida yang akan memproses zat dan kuman tersebut dapat dilihat pada gambar 4.37.

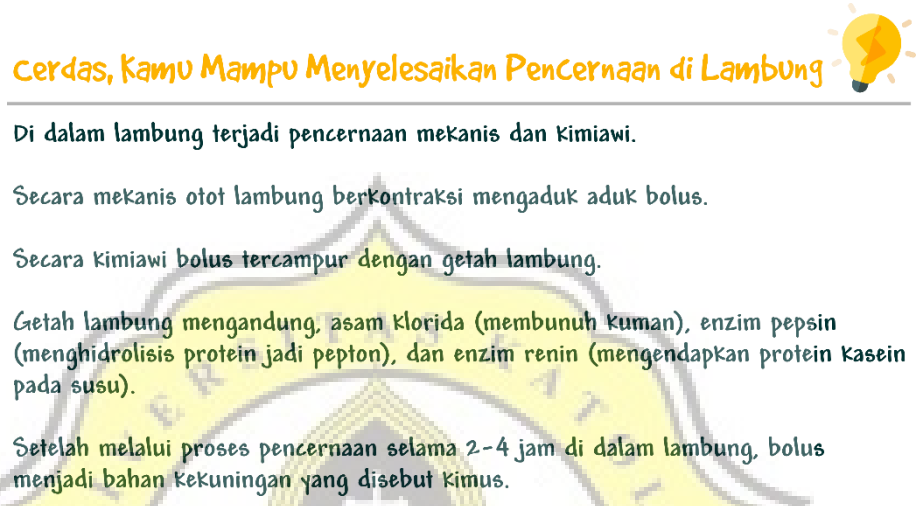


Gambar 4.35 Instruksi Stage 3

Gambar 4.36 Protein, Kuman, dan Protein Kasein

Gambar 4.37 Kapsul Enzim Pepsin, Asam Klorida, dan Enzim Renin

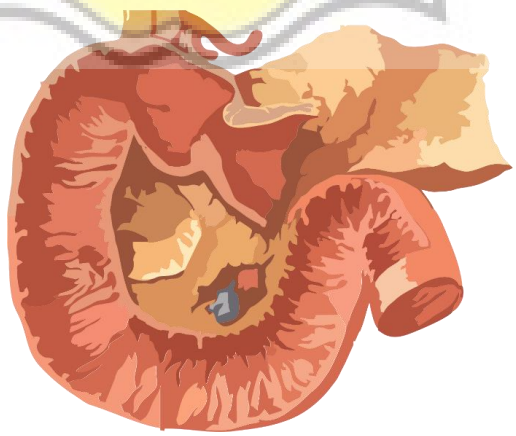
Setelah pemain telah memasukkan zat makanan yang benar ke dalam kapsul enzim dan asam klorida sesuai target, maka akan muncul tampilan informasi berkaitan dengan pencernaan di lambung seperti pada gambar 4.38.



Gambar 4.38 Tampilan Informasi Stage 3

4.2.5 Stage 4

Pada *stage* 4, pencernaan dilakukan di duodenum, oleh karena itu tampilan desain *gameplay* menggunakan tampilan duodenum seperti pada gambar 4.39 dan disertai dengan tampilan kimus seperti pada gambar 4.40.



Gambar 4.39 Duodenum

Gambar 4.40 Kimus

Di *stage* 4 ini, pemain harus mengarahkan zat yang berasal dari kimus menuju ke enzim-enzim yang akan memproses zat tersebut. Instruksi tersebut dapat dilihat pada gambar 4.41. Tampilan untuk amilum dan lemak dapat dilihat pada gambar 4.42. Kemudian untuk tampilan enzim-enzim yang akan memproses zat tersebut dapat dilihat pada gambar 4.43.

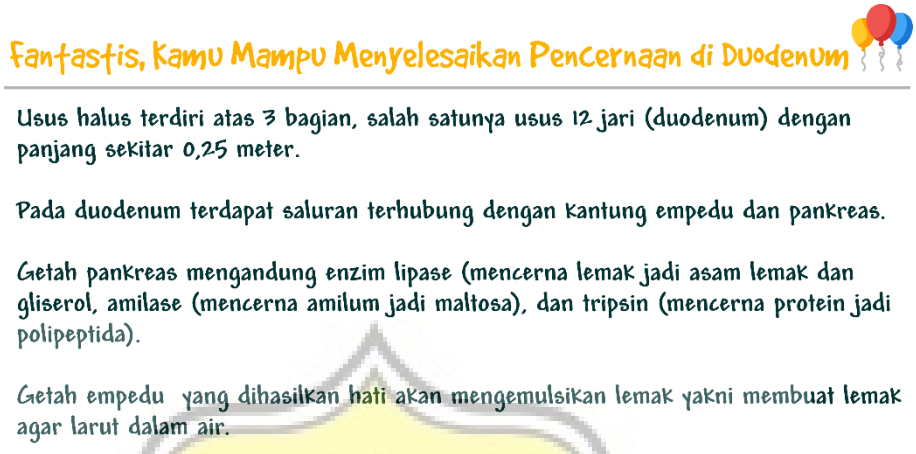
Gambar 4.41 Instruksi Stage 4

Gambar 4.42 Amilum dan Lemak

Gambar 4.43 Kapsul Enzim Lipase, Enzim Tripsin, dan Enzim Amilase

Setelah pemain telah memasukkan zat makanan yang benar ke dalam kapsul enzim sesuai target, maka akan muncul tampilan

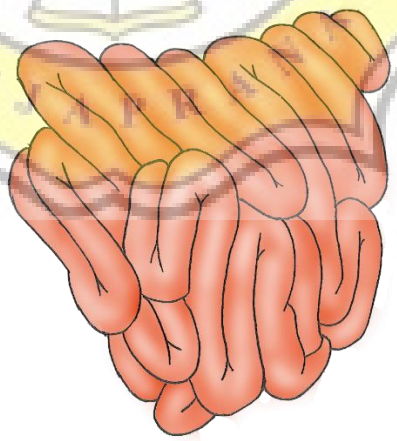
informasi berkaitan dengan pencernaan di duodenum seperti pada gambar 4.44.



Gambar 4.44 Tampilan Informasi Stage 4

4.2.6 Stage 5

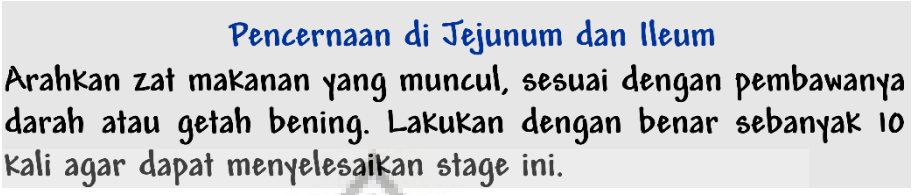
Pada *stage* 5, pencernaan dilakukan di usus halus yaitu jejunum dan ileum, oleh karena itu tampilan desain *gameplay* menggunakan tampilan jejunum dan ileum seperti pada gambar 4.45 dan disertai dengan tampilan kimus seperti pada gambar 4.40.



Gambar 4.45 Jejunum dan Ileum

Di *stage* 5 ini, pemain harus mengarahkan zat yang berasal dari kimus menuju ke darah atau getah bening yang akan membawa zat

tersebut. Instruksi tersebut dapat dilihat pada gambar 4.46. Tampilan untuk zat-zat makanan yang muncul pada *stage* 5 ini dapat dilihat pada gambar 4.47. Kemudian untuk tampilan darah dan getah bening yang akan membawa zat tersebut dapat dilihat pada gambar 4.48.

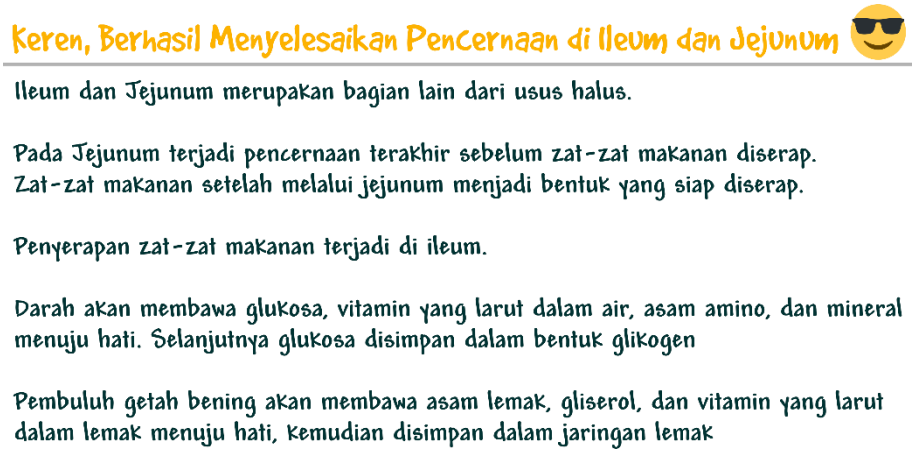


Gambar 4.46 Instruksi Stage 5

Gambar 4.47 Zat-zat Makanan Stage 5

Gambar 4.48 Darah dan Getah Bening

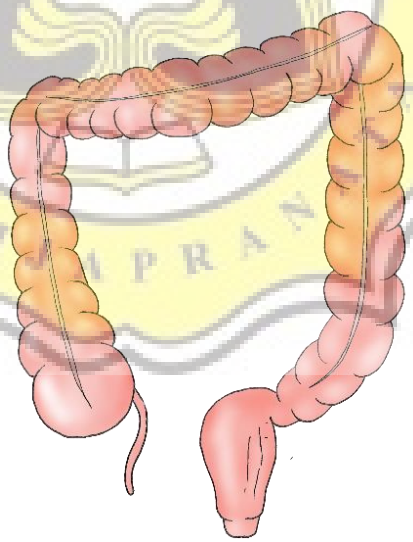
Setelah pemain telah memasukkan zat makanan yang benar ke dalam darah atau getah bening sesuai target, maka akan muncul tampilan informasi berkaitan dengan pencernaan di jejunum dan ileum seperti pada gambar 4.49.



Gambar 4.49 Tampilan Informasi Stage 5

4.2.7 Stage 6

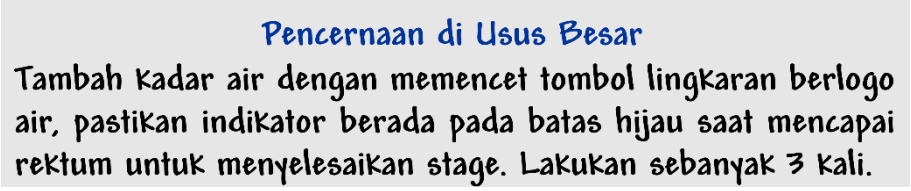
Pada *stage* 6, pencernaan dilakukan di usus besar, oleh karena itu tampilan desain *gameplay* menggunakan tampilan usus besar seperti pada gambar 4.50 dan disertai dengan tampilan sisa makanan seperti pada gambar 4.51.



Gambar 4.50 Usus Besar

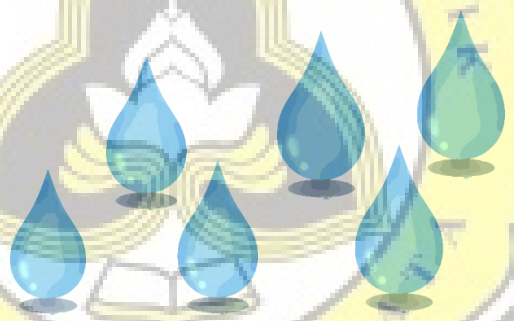
Gambar 4.51 Sisa Makanan Padat dan Sisa Makanan Cair

Di *stage* 6 ini, pemain harus mengatur kadar air pada sisa makanan agar dapat menyelesaikan *stage*. Instruksi yang harus diikuti pemain dapat dilihat pada gambar 4.52.



Gambar 4.52 Instruksi Stage 6

Gambar 4.53 merupakan gambaran air yang keluar dari sisa makanan dalam proses pencernaan pada usus besar, semakin banyak jumlah tetesan maka indikator air pada gambar 4.54 akan berkurang semakin banyak pula.



Gambar 4.53 Air

Gambar 4.54 Indikator Air

Untuk menambah jumlah air pada sisa makanan, pemain harus menekan tombol air. Tampilan dari tombol air dapat dilihat pada gambar 4.55.

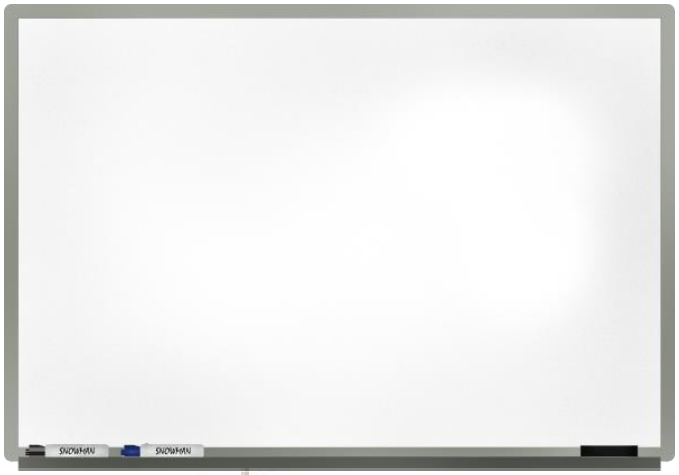
Gambar 4.55 Tombol Air

Setelah sisa makanan berhasil menuju rektum dengan kadar air yang sesuai, maka akan muncul tampilan informasi yang berkaitan dengan proses pencernaan di usus besar seperti pada gambar 4.56.

Gambar 4.56 Tampilan Informasi Stage 6

4.2.8 Bagian Informasi

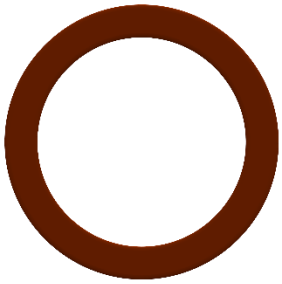
Pada setiap bagian *gameplay* menggunakan background papan tulis seperti pada gambar 4.57 begitu pula bagian informasi pada setiap *stage*.



Gambar 4.57 Background Papan Tulis

Pada bagian informasi terdapat tombol *next* untuk melanjutkan ke *stage* berikutnya. Sekaligus kursor berbentuk tangan petunjuk yang digunakan untuk memencet tombol tersebut. Tampilan dari kursor dan tombol dapat dilihat pada gambar 4.58. Ketika kursor diarahkan pada tombol *next* tersebut maka akan muncul *progress bar* berbentuk lingkaran seperti pada gambar 4.59 yang menunjukkan *progress* untuk perpindahan *stage*, jika sebelum *progresss bar* penuh kursor dipindahkan dari tombol *next* maka pemain harus memencet tombol lagi dan *progress* dimulai dari awal lagi.

Gambar 4.58 Kursor dan Tombol Next



Gambar 4.59 Progress Bar Lingkaran

Setiap perpindahan bagian atau *stage* dalam *game* akan diawali dengan tampilan *loading* seperti pada gambar 4.60 sebelum akhirnya menuju *stage* berikutnya atau bagian tertentu pada *game*.

Gambar 4.60 Tampilan Loading

4.2.9 Credit

Tampilan yang terdapat pada bagian *credit*, seperti pada gambar 4.61. Dengan menggunakan kursor seperti pada gambar 4.58, dan tombol *back* yang terdapat pada *credit* seperti gambar 4.62.

Gambar 4.61 Tampilan Credit

Gambar 4.62 Tombol Back

4.2.10 Game Over

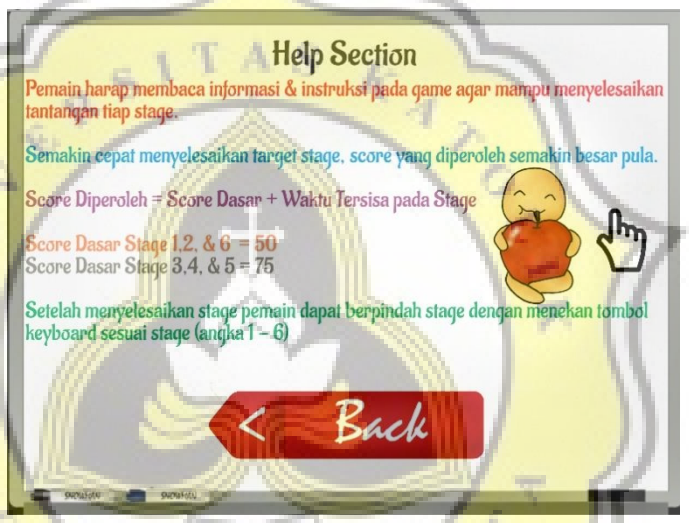
Tampilan yang terdapat pada bagian *game over*, seperti pada gambar 4.63. Turut menampilkan *score* terakhir yang diperoleh pemain. Kemudian tampilan tombol retry dan main menu dapat dilihat pada gambar 4.64.

Gambar 4.63 Tampilan Game Over

Gambar 4.64 Tombol Retry dan Main Menu

4.2.11 Help Section

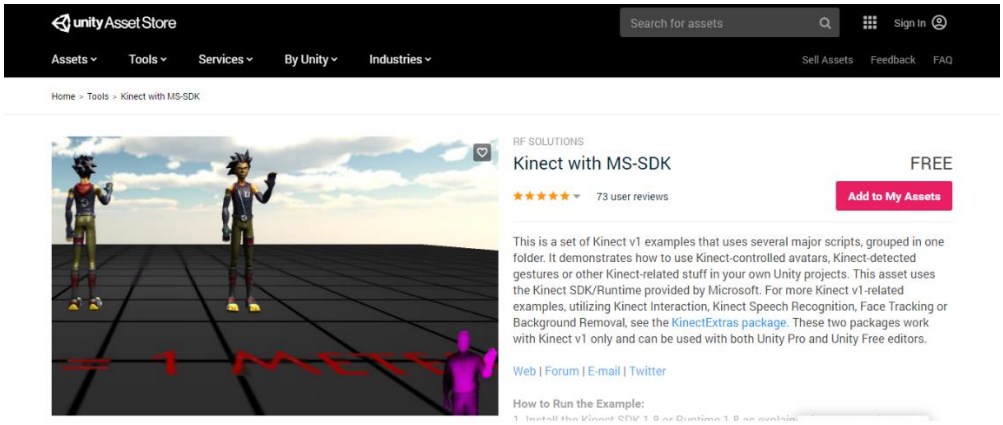
Tampilan yang terdapat pada bagian *help section*, seperti pada gambar 4.65. Menampilkan informasi bagi pemain mengenai perolehan score dan fitur perpindahan stage menggunakan tombol keyboard jika telah menyelesaikan stage tersebut.



Gambar 4.65 Tampilan Help Section

4.3 Pemrograman Game "Mari Mencerna"

Pemrograman *game* "Mari Mencerna" menggunakan *game engine* Unity versi 5.6.3p2 dan menggunakan bahasa pemrograman C#. Untuk dapat menghubungkan unity dengan Kinect menggunakan *plugin* tambahan dari asset store Unity, yaitu Kinect with MS-SDK yang dikembangkan oleh RF Solutions seperti pada gambar 4.66.



Gambar 4.66 Tampilan plugin Kinect with MS-SDK

4.3.1 Script Tombol

Script pada gambar 4.67 ini digunakan untuk mengatur tombol perpindahan *stage*, sehingga saat kursor digerakan mengenai tombol, maka akan muncul *bar* berbentuk lingkaran yang akan terisi dan akan melakukan perpindahan *stage* jika telah penuh. Saat perpindahan *stage* akan muncul gambar yang menunjukkan bahwa perpindahan *stage* sedang *loading*.

```
public class Button : MonoBehaviour {

    private RectTransform canvasRectT,waitBar;
    public GameObject myCanvas,theCursor,loading,hide;
    private Transform objectToFollow;
    public Image bar;
    public string indicator;
    float progress;
    public AudioClip audioButton;
    private AudioSource asClicked;
    private static string stage;
    public static string theStage
    {
        get
        {
            return stage;
        }
        set
        {
            stage = value;
        }
    }

    void Start () {
```

```
        canvasRectT = myCanvas.GetComponent<RectTransform> ();
        waitBar = bar.GetComponent<RectTransform>();
        objectToFollow = theCursor.GetComponent<Transform>();
        progress = 0;
        BarChange (progress);
        asClicked = gameObject.AddComponent<AudioSource>();
        asClicked.clip = audioButton;
        asClicked.volume = 0.3f;
    }
    void Update () {
        Vector2 screenPoint = RectTransformUtility.WorldToScreenPoint(Came
ra.main, objectToFollow.position);
        waitBar.anchoredPosition = screenPoint - canvasRectT.sizeDelta / 2f;
    }
    void OnTriggerExit2D(Collider2D collider){
        progress = 0;
        BarChange (progress);
        asClicked.Stop ();
    }
    //wait bar to change stage
    void OnTriggerEnter2D(Collider2D collider){
        asClicked.Play ();
    }
    void OnTriggerStay2D(Collider2D collider){
        float increment = 0.01f;
        if (progress <= 1) {
            progress += increment;
            BarChange (progress);
        }
        if (progress >= 0.99) {
            Destroy(theCursor);
            hide.SetActive (false);
            loading.SetActive(true);
            if (indicator.Equals ("Quit")) {
                Application.Quit ();
                print ("QUIT");
            } else if (indicator.Equals ("Credit")) {
                ChangeStage ("Credit");
            } else if (indicator.Equals ("Retry")) {
                ChangeStage (stage);
            } else {
                ChangeStage (collider.name);
            }
        }
    }
    void BarChange(float time)
    {
        bar.fillAmount = time;
    }
    //move to another stage
    void ChangeStage(string stagename)
    {
        if(stagename.Equals("MainMenu")){
            LimitScore.Scored = 0;
        }
        this.gameObject.SetActive (false);
        Destroy (myCanvas);
        SceneManager.LoadScene(stagename, LoadSceneMode.Single);
```

```
    }
}
```

Gambar 4.67 Script Button.cs

4.3.2 Script Pergerakan

Script ini digunakan untuk mengatur pergerakan *game object* sesuai dengan pergerakan pemain. Dapat dilihat pada gambar 4.68.

```
public class MoveTowards : MonoBehaviour {

    public GameObject Followed;
    float speed;

    void Update () {
        float step = speed * Time.deltaTime;
        this.gameObject.transform.position = Vector2.MoveTowards(Followed.t
ransform.position, Followed.transform.position, step);
    }
}
```

Gambar 4.68 Script MoveTowards.cs

4.3.3 Script Stage 1

Script pada gambar 4.69 ini digunakan untuk mengatur *gameplay* pada *stage* 1. Mulai dari mengatur pergerakan animasi rahang dan apel, hingga penghitungan *score* yang menjadi syarat untuk berpindah ke *stage* berikutnya.

```
public class Stage1 : MonoBehaviour {

    public GameObject atas,bawah,atasPointer,bawahPointer;
    public GameObject theCursor,nextButton,gameplay,info;
    public GameObject Information,Apple;
    public GUIText targetText,scoreText;
    public float time;
    public Sprite apple1,apple2,apple3,apple4;
    SpriteRenderer sr;
    float atasx,atasy,bawahx,bawahy;
    int gigit = 5;
    public AudioClip audioChew;
    private AudioSource asChew;

    void Start () {
        Timer.timeLeft = time;
        asChew = gameObject.AddComponent<AudioSource>();
        asChew.clip = audioChew;
        sr = Apple.GetComponent<SpriteRenderer>();
        PhaseCheck.Check = 1;
        atasx = atas.transform.position.x;
```

```
        atasy = atas.transform.position.y;
        bawahx = bawah.transform.position.x;
        bawahy = bawah.transform.position.y;
    }
    void Update () {
        targetText.text = ": " + gigit.ToString();
        scoreText.text = "Score: " + LimitScore.Scored;
        if (gigit == 4) {
            sr.sprite = apple1;
        }
        if (gigit == 3) {
            sr.sprite = apple2;
        }
        if (gigit == 2) {
            sr.sprite = apple3;
        }
        if (gigit == 1) {
            sr.sprite = apple4;
        }
        if (gigit <= 0) {
            StartCoroutine (Wait ());

        }
    }
    void OnTriggerEnter2D(Collider2D collider)
    {
        if (PhaseCheck.Check == 1) {
            if (atasPointer.transform.position.y > bawahPointer.transform.position
.y) {
                asChew.Play ();
                int point = 50 + Mathf.RoundToInt (Timer.timeLeft);
                LimitScore.Scored += point;
                atas.transform.Translate (0, Time.deltaTime - 0.27f, 0);
                bawah.transform.Translate (0, Time.deltaTime + 0.27f, 0);
                StartCoroutine (Example ());
                gigit--;
                PhaseCheck.Check = 0;
            }
        }
    }
    IEnumerator Example()
    {
        yield return new WaitForSeconds(0.45f);
        SetDefaultPos ();
    }
    IEnumerator Wait()
    {
        yield return new WaitForSeconds(0.4f);
        info.SetActive (true);
        atasPointer.SetActive (false);
        nextButton.SetActive (true);
        gameplay.SetActive (false);
        bawahPointer.SetActive (false);
        theCursor.GetComponent<SpriteRenderer> ().enabled = true;
        Information.SetActive (true);
        PlayerPrefs.SetInt ("Stage", 1);
    }
    void SetDefaultPos(){
```

```
        atas.transform.position = new Vector3(atasx, atasy, 0);
        bawah.transform.position = new Vector3(bawahx, bawahy,0);
        PhaseCheck.Check = 1;
    }
```

Gambar 4.69 Script Stage1.cs

4.3.4 Script Deteksi Sentuhan Stage 2

Script ini digunakan untuk mendeteksi sentuhan pada sebuah objek yang terdapat di *stage* 2. Dapat diamati pada gambar 4.70.

```
public class TouchStage2 : MonoBehaviour {

    void OnTriggerEnter2D(Collider2D collider)
    {
        if (PhaseCheck.Check == 1) {
            if (collider.tag == "left") {
                TouchDetected.Left = true;
            }
            if (collider.tag == "right") {
                TouchDetected.Right = true;
            }
        }
    }
    void OnTriggerExit2D(Collider2D collider)
    {
        if (collider.tag == "left") {
            TouchDetected.Left = false;
        }
        if (collider.tag == "right") {
            TouchDetected.Right = false;
        }
    }
}
```

Gambar 4.70 Script TouchStage2.cs

4.3.5 Script Stage2

Script ini digunakan untuk mengatur *gameplay* pada *stage* 2. Mulai dari mengatur pergerakan bolus dan *icon* untuk sentuhan tangan. Sekaligus untuk mengaktifkan dan menonaktifkan *icon* sentuhan tangan untuk sementara. Selain itu *script* pada gambar 4.71 ini juga megatur syarat tantangan untuk *stage* 2.

```
public class Stage2 : MonoBehaviour {

    public GameObject gameplay,info,Information;
    public GameObject Esofagus,lhand,rhand,ltouch,rtouch;
```

```
public GameObject theCursor,nextButton;
int sentuh = 5;
public Sprite eso1,eso2,eso3,eso4;
SpriteRenderer sr;
float leftx,lefty,rightx,righty;
public AudioClip audioButton;
private AudioSource asClicked;
public GUIText targetText,scoreText;
public float time;

void Start () {
    Timer.timeLeft = time;
    asClicked = gameObject.AddComponent<AudioSource>();
    asClicked.clip = audioButton;
    sr = Esofagus.GetComponent<SpriteRenderer>();
    leftx = ltouch.transform.position.x;
    lefty = ltouch.transform.position.y;
    rightx = rtouch.transform.position.x;
    righty = rtouch.transform.position.y;
    PhaseCheck.Check = 1;
    print ("awalcek");
}
void Update () {
    targetText.text = ": " + sentuh.ToString();
    scoreText.text = "Score: " + LimitScore.Scored;
    if (sentuh == 4) {
        sr.sprite = eso1;
        ltouch.transform.position = new Vector3 (leftx, lefty, 0);
        rtouch.transform.position = new Vector3 (rightx, righty, 0);
        PhaseCheck.Check = 1;
        ActiveCollider ();
    }
    if (sentuh == 3) {
        sr.sprite = eso2;
        ltouch.transform.position = new Vector3 (leftx, lefty, 0);
        rtouch.transform.position = new Vector3 (rightx, righty, 0);
        PhaseCheck.Check = 1;
        ActiveCollider ();
    }
    if (sentuh == 2) {
        sr.sprite = eso3;
        ltouch.transform.position = new Vector3 (leftx, lefty, 0);
        rtouch.transform.position = new Vector3 (rightx, righty, 0);
        PhaseCheck.Check = 1;
        ActiveCollider ();
    }
    if (sentuh == 1) {
        sr.sprite = eso4;
        ltouch.transform.position = new Vector3 (leftx, lefty, 0);
        rtouch.transform.position = new Vector3 (rightx, righty, 0);
        PhaseCheck.Check = 1;
        ActiveCollider ();
    }
    if (sentuh <= 0) {
        StartCoroutine (Wait ());
    }
    TouchChecker();
}
```

```
void TouchChecker(){
    if(TouchDetected.Left==true && TouchDetected.Right==true){
        asClicked.Play ();
        int point = 50 + Mathf.RoundToInt (Timer.timeLeft);
        LimitScore.Scored += point;
        DeactiveCollider ();
        sentuh--;
        righty+=-1.38f;
        lefty +=-1.38f;
        if (sentuh == 4 || sentuh == 2) {
            rightx += 2.2f;
            leftx -= 2.2f;
        }
        if (sentuh == 3 || sentuh == 1) {
            rightx -= 2.2f;
            leftx += 2.2f;
        }
        PhaseCheck.Check = 0;
    }

}
IEnumerator Wait()
{
    yield return new WaitForSeconds (0.4f);
    info.SetActive (true);
    PhaseCheck.Check = 0;
    nextButton.SetActive (true);
    gameplay.SetActive(false);
    theCursor.GetComponent<SpriteRenderer> ().enabled = true;
    Information.SetActive (true);
    PlayerPrefs.SetInt ("Stage", 2);
}
void ActiveCollider(){
    lhand.GetComponent<PolygonCollider2D> ().enabled = true;
    rhand.GetComponent<PolygonCollider2D> ().enabled = true;
    PhaseCheck.Check = 1;
}
void DeactiveCollider(){
    lhand.GetComponent<PolygonCollider2D> ().enabled = false;
    rhand.GetComponent<PolygonCollider2D> ().enabled = false;
}
}
```

Gambar 4.71 Script Stage2.cs

4.3.6 Script Tangan Stage 3,4 dan 5

Script pada gambar 4.72 ini digunakan untuk dapat menangkap objek lain dan melakukan *drag* terhadap objek tersebut.

```
public class Hand : MonoBehaviour {

    float speed=1f;
    private bool following;
    void Start () {
```

```
        following = false;
    }
    void OnTriggerStay2D(Collider2D collider)
    {

        if (collider.gameObject.CompareTag("Drag"))
            {
            if (following)
            {
                DragCheck.Check = 1;
                following = false;
            }
            else
            {
                following = true;
            }
            }
        if (following)
        {
            float step = speed * Time.deltaTime;
            if (collider.gameObject.CompareTag ("Drag")) {
                collider.gameObject.transform.position = Vector2.MoveTowards (t
his.transform.position, this.transform.position, step);
            }
        }
    }
}
```

Gambar 4.72 Script Hand.cs

4.3.7 Script Kapsul Stage 3 dan 4

Script ini digunakan untuk mendeteksi objek yang bersentuhan dengan kapsul. Bila objek sesuai maka object akan dihancurkan dan muncul suara yang menandakan benar. Bila objek tidak sesuai setelah objek dihancurkan muncul suara yang menandakan salah. *Script* lengkapnya dapat dilihat pada gambar 4.73.

```
public class Capsule : MonoBehaviour {

    public AudioClip audioCorrect,audioWrong;
    private AudioSource asCorrect,asWrong;
    private int cek;
    public GameObject detect;
    public GUIText scoreText;
    void Start()
    {
        asCorrect = gameObject.AddComponent<AudioSource>();
        asCorrect.clip = audioCorrect;
        asWrong = gameObject.AddComponent<AudioSource>();
        asWrong.clip = audioWrong;
    }
```

```
void Update()
{
    scoreText.text = "Score: " + LimitScore.Scored;
    cek = DragCheck.Check;
}
void OnTriggerEnter2D(Collider2D collider)
{
    print ("trig");
    if (cek == 1)           {
        Debug.Log(" c " + collider.name + " & " + detect.name);
        if (collider.gameObject.CompareTag ("Drag")) {
            Destroy (collider.gameObject);
            if (collider.name.Contains (detect.name)) {
                int point = 80 + Mathf.RoundToInt (Timer.timeLeft);
                LimitScore.Scored += point;
                asCorrect.Play ();
                LimitScore.Limit -= 1;
            } else {
                asWrong.Play ();
            }
            DragCheck.Check = 0;
            PhaseCheck.Check = 1;
        }
    }
}
}
```

Gambar 4.73 Script Capsule.cs

4.3.8 Script Spawner Objek Stage 3 dan 4

Script gambar 4.74 ini digunakan untuk memunculkan objek untuk *stage* 3 dan *stage* 4 dengan diiringi suara.

```
public class SpawnerS34 : MonoBehaviour {

    public GameObject item1,item2,item3;
    public float posx,posy;
    public AudioClip audioSpawn;
    private AudioSource asSpawn;

    void Start () {
        asSpawn = gameObject.AddComponent<AudioSource>();
        asSpawn.clip = audioSpawn;
    }

    void Update () {
        if (Score.Scored > 0) {
            if (PhaseCheck.Check == 1) {
                Spawn ();
                PhaseCheck.Check = 0;
            }
        }
    }
    void Spawn(){
        var a = Random.Range(0, 3);
```

```
        asSpawn.Play ();
        if (a == 0) {
            GameObject.Instantiate (item1, new Vector3 (posx, posy, 0), Quatern
ion.identity);
        } else if (a == 1) {
            GameObject.Instantiate (item2, new Vector3 (posx,  posy, 0), Quater
nion.identity);
        } else if (a == 2) {
            GameObject.Instantiate (item3, new Vector3 (posx,  posy, 0), Quater
nion.identity);
        }
    }
}
```

Gambar 4.74 Script SpawnerS34.cs

4.3.9 Script Stage 3

Script pada gambar 4.75 ini digunakan untuk mengatur *gameplay* pada *stage* 3. Pada *script* ini diatur syarat *score* yang dibutuhkan untuk berpindah *stage*.

```
public class Stage3 : MonoBehaviour {

    public GameObject gameplay,info,Information;
    public GameObject bolus,lambung,theCursor,nextButton;
    public GUIText targetText;
    public float time;
    void Start () {
        Timer.timeLeft = time;
        LimitScore.Limit = 5;
    }

    // Update is called once per frame
    void Update () {
        targetText.text= ": "+LimitScore.Limit.ToString();
        if (LimitScore.Limit <=0) {
            StartCoroutine (Wait ());

        }
    }
    IEnumerator Wait()
    {
        yield return new WaitForSeconds (0.4f);
        PhaseCheck.Check = 0;
        info.SetActive (true);
        Information.SetActive (true);
        nextButton.SetActive (true);
        gameplay.SetActive(false);
        bolus.SetActive (false);
        lambung.SetActive(false);
        targetText.enabled = false;
        theCursor.GetComponent<SpriteRenderer> ().enabled = true;
        PlayerPrefs.SetInt ("Stage", 3);
```

```
    }
}
```

Gambar 4.75 Script Stage3.cs

4.3.10 Script Stage 4

Pada *script* gambar 4.76 ini dilakukan pengaturan *gameplay* untuk *stage* 4. Pada *script* ini diatur syarat *score* yang dibutuhkan untuk berpindah *stage*.

```
public class Stage4 : MonoBehaviour {

    public GameObject gameplay,info,Information;
    public GameObject nextButton,theCursor,kimus;
    public GUIText targetText;
    public float time;

    void Start () {
        Timer.timeLeft = time;
        LimitScore.Limit = 7;
    }
    void Update () {
        targetText.text = ": " + LimitScore.Limit.ToString();
        print ("pembatas");
        print (PhaseCheck.Check);
        if (LimitScore.Limit <=0) {
            StartCoroutine (Wait ());
        }
    }
    IEnumerator Wait()
    {
        yield return new WaitForSeconds (0.4f);
        PhaseCheck.Check = 0;
        info.SetActive (true);
        nextButton.SetActive (true);
        gameplay.SetActive(false);
        Information.SetActive (true);
        targetText.enabled = false;
        theCursor.GetComponent<SpriteRenderer> ().enabled = true;
        kimus.SetActive (false);
        PlayerPrefs.SetInt ("Stage", 4);
    }
}
```

Gambar 4.76 Script Stage4.cs

4.3.11 Script Spawner Objek Stage 5

Script gambar 4.77 ini digunakan untuk memunculkan objek untuk *stage* 5 secara acak berdasarkan *list* yang ada.

```
public class SpawnerS5 : MonoBehaviour {
```

```
    public GameObject itemdarah,itemgetah;
    public Sprite vitA,vitB,vitC,vitD,vitE,vitK,gliserol,glukosa,mineral,amino,asa
mlemak;
    List<int> mylist;
    int a,i;
    SpriteRenderer sr;

    void Start () {
        mylist = new List<int>();   //  Declare list
        for (int n = 0; n < 10; n++)    //  Populate list
        (  mylist.Add(n); }
        a = Random.Range(0,mylist.Count+1);
    }
    void Update () {
        if (mylist.Count == 0&&PhaseCheck.Check==1) {
            for (int n = 0; n < 10; n++)    //  Populate list
            {
                mylist.Add(n);
            }
        }
        if (Score.Scored > 0) {
            if (PhaseCheck.Check == 1) {
                a = Random.Range (0, mylist.Count + 1);
                Spawn ();
                PhaseCheck.Check = 0;
            }
        }
    }
    void Spawn(){
        //  Pick random element from the list
        i = mylist[a];    //  i = the number that was randomly picked
        mylist.RemoveAt(a);   //  Remove chosen element
        if (i == 0) {
            Darah (glukosa);
        } else if (i == 1) {
            Darah (vitB);
        } else if (i == 2) {
            Darah (vitC);
        } else if (i == 3) {
            Darah (amino);
        }  else if (i == 4) {
            Darah (mineral);
        } else if (i == 5) {
            Getah (vitA);
        } else if (i == 6) {
            Getah (vitD);
        } else if (i == 7) {
            Getah (vitE);
        } else if (i == 8) {
            Getah (vitK);
        } else if (i == 9) {
            Getah (asamlemak);
        } else if (i == 10) {
            Getah (gliserol);
        }
    }
    void Darah (Sprite theSprite){
        GameObject.Instantiate (itemdarah, new Vector3 (-
```

```
1.59f, 1.91f, 0), Quaternion.identity);
        sr = itemdarah.GetComponent<SpriteRenderer>();
        sr.sprite = theSprite;
        print("darah");
    }
    void Getah (Sprite theSprite){
        GameObject.Instantiate (itemgetah, new Vector3 (-
1.59f, 1.91f, 0), Quaternion.identity);
        sr = itemgetah.GetComponent<SpriteRenderer>();
        sr.sprite = theSprite;
        print("getah");
    }
```

Gambar 4.77 Script SpawnerS5.cs

4.3.12 Script Stage 5

Script pada gambar 4.78 ini digunakan untuk mengatur gameplay Stage 5.

```
public class Stage5 : MonoBehaviour {

    public GameObject gameplay, info, Information;
    public GameObject nextButton, theCursor, UsusHalus;
    public GUIText targetText;
    public float time;

    void Start () {
        Timer.timeLeft = time;
        LimitScore.Limit = 10;
    }

    void Update () {
        targetText.text = ": " + LimitScore.Limit.ToString();
        if (LimitScore.Limit <=0) {
            StartCoroutine (Wait ());
        }
    }
    IEnumerator Wait()
    {
        yield return new WaitForSeconds (0.4f);
        PhaseCheck.Check = 0;
        info.SetActive (true);
        nextButton.SetActive (true);
        gameplay.SetActive(false);
        targetText.enabled = false;
        Information.SetActive (true);
        UsusHalus.SetActive (false);
        theCursor.GetComponent<SpriteRenderer> ().enabled = true;
        PlayerPrefs.SetInt ("Stage", 5);
    }
}
```

Gambar 4.78 Script Stage5.cs

4.3.13 Script Tombol Air Stage 6

Script ini digunakan untuk mengatur tombol pengaturan air untuk *stage* 6, sehingga saat ditekan indikator air akan bertambah sekaligus mengeluarkan suara. Berikut pemaparan *script* pada gambar 4.79.

```
public class WaterButton : MonoBehaviour {

    public Image bar;
    float progress;
    public int cek,ganti;
    public AudioClip waterButton;
    private AudioSource waterButtonPress;
    SpriteRenderer mySR;

    void Start () {
        waterButtonPress = gameObject.AddComponent<AudioSource>();
        waterButtonPress.clip = waterButton;
        mySR = this.gameObject.GetComponent<SpriteRenderer>();
    }

    void Update () {
        if (PhaseCheck.Check == cek) {
            mySR.color = Color.white;
        } else {
            mySR.color = Color.grey;
        }
        if (Input.GetKeyDown(KeyCode.Space))
        {
            print (PhaseCheck.Check);
            if (PhaseCheck.Check == cek) {
                waterButtonPress.Play ();
                waterButtonPress.volume = 0.5f;
                BarChange (bar.fillAmount + 0.1f);
                PhaseCheck.Check = ganti;
            }
        }
    }
    void BarChange(float amount)
    {
        bar.fillAmount = amount;
    }
    void OnTriggerEnter2D(Collider2D collider)
    {
        print (PhaseCheck.Check);
        if (PhaseCheck.Check == cek) {
            waterButtonPress.Play ();
            waterButtonPress.volume = 0.5f;
            BarChange (bar.fillAmount + 0.1f);
            PhaseCheck.Check = ganti;
        }
```

```
    }
}
```

Gambar 4.79 Script WaterButton.cs

4.3.14 Script Spawner Air

Script pada gambar 4.80 ini digunakan untuk memunculkan air secara *random* dan mengurangi bar indikator air.

```
public class WaterSpawn : MonoBehaviour {

    public Image bar;
    public GameObject item1, item2, item3, kotoran;
    float posx,posy;
    float progress;
    public AudioClip audioSuccess,audioFailed,  waterDrop;
    private AudioSource asSuccess, asFailed, waterSound;
    private Animator anim;
    // Use this for initialization
    void Start () {
        asSuccess = gameObject.AddComponent<AudioSource>();
        asSuccess.clip = audioSuccess;
        asFailed = gameObject.AddComponent<AudioSource>();
        asFailed.clip = audioFailed;
        waterSound = gameObject.AddComponent<AudioSource>();
        waterSound.clip = waterDrop;
        anim = GetComponent<Animator>();
    }
    // Update is called once per frame
    void Update () {
        posx = kotoran.transform.position.x;
        posy = kotoran.transform.position.y;
        if (bar.fillAmount < 0.32f) {
            anim.SetBool ("keras", true);
        }
        if (bar.fillAmount > 0.32f) {
            anim.SetBool ("keras", false);
        }
    }
    void Spawn(){
        waterSound.Play ();
        var a = Random.Range(0, 3);
        print (a);
        if (a == 0) {
            progress = bar.fillAmount - 0.04f;
            GameObject.Instantiate (item1, new Vector3 (posx, posy, 0), Quatern
ion.identity);
        } else if (a == 1) {
            progress = bar.fillAmount - 0.13f;
            GameObject.Instantiate (item2, new Vector3 (posx,  posy, 0), Quater
nion.identity);
        } else if (a == 2) {
            progress = bar.fillAmount - 0.22f;
            GameObject.Instantiate (item3, new Vector3 (posx,  posy, 0), Quater
nion.identity);
```

```
        }
        BarChange (progress);
    }
    void BarChange(float amount)
    {
        bar.fillAmount = amount;
    }
    void CekBerhasil()
    {
        if (bar.fillAmount > 0.32f && bar.fillAmount < 0.68f) {
            asSuccess.Play ();
            Score.Scored -= 1;
        } else {
            asFailed.Play ();
        }
    }
```

Gambar 4.80 Script WaterSpawn.cs

4.3.15 Script Water Drop

Script gambar 4.81 ini digunakan membuat air yang telah muncul jatuh kebawah kemudian menghilang.

```
public class WaterDrop : MonoBehaviour {

    float posy;
    float time = 1f;

    void Start () {
        posy = this.transform.position.y;
    }

    void Update () {
        posy = this.transform.position.y - 0.05f;
        this.transform.position = new Vector3 (this.transform.position.x, posy, 0)
;
        time -= Time.deltaTime;
        if (time < 0) {
            Destroy (this.gameObject);
        }
    }
}
```

Gambar 4.81 Script WaterDrop.cs

4.3.16 Script Stage 6

Pada *script* gambar 4.82 ini *gameplay* untuk *Stage* 6 diatur.

```
public class Stage6 : MonoBehaviour {

    public GameObject gameplay,info,Information;
    public GameObject nextButton,theCursor,kotoran;
    public GUIText targetText;
    public float time;
```

```
void Start () {
    Timer.timeLeft = time;
    LimitScore.Limit = 3;
    PhaseCheck.Check = 1;
}

void Update () {
    targetText.text = ": "+LimitScore.Limit.ToString();
    if (LimitScore.Limit <=0) {
        Wait();
    }
}
void FinishStage()
{
    Information.SetActive (true);
    PhaseCheck.Check = 0;
    info.SetActive (true);
    nextButton.SetActive (true);
    gameplay.SetActive(false);
    Destroy (kotoran);
    theCursor.GetComponent<SpriteRenderer> ().enabled = true;
    PlayerPrefs.SetInt ("Stage", 6);
}
void Wait()
{
    float f = 0;
    if (f < 1) {
        f++;
    }
    if(f==1)
    {
        FinishStage ();
    }
}
}
```

Gambar 4.82 Script Stage6.cs

4.3.17 Script Drag

Script gambar 4.83 ini memungkinkan pemain melakukan *drag* pada objek yang disentuhnya.

```
public class DragDrop : MonoBehaviour {
    public float moveSpeed;
    public float offset = 0.05f;
    private bool following;
    // Use this for initialization
    void Start () {
        following = false;
        offset += 10;
    }

  // Update is called once per frame
  void Update () {
        if (Input.GetMouseButtonDown(0) && ((Camera.main.ScreenToWorldP
```

```
oint(Input.mousePosition) - transform.position).magnitude <= offset))
        {
            if (following)
            {
                DragCheck.Check = 1;
                print ("false");
                following = false;
            }
            else
            {
                following = true;
            }
        }
        if (following)
        {
            transform.position = Vector2.Lerp(transform.position, Camera.main.S
creenToWorldPoint(Input.mousePosition), moveSpeed);
        }
    }
}
```

Gambar 4.83 Script DragDrop.cs

4.3.18 Script Tampilan Awal

Script gambar 4.84 mengatur tampilan awal yang muncul pada tiap stage pada game ini.

```
public class OpeningStage : MonoBehaviour {

    // Use this for initialization
    public GameObject gameplay;
    public GameObject stageObject;
    public float time;

    void Update () {
        time -= Time.deltaTime;
        if (time < 0)
        {
            gameplay.SetActive (true);
            stageObject.SetActive(true);
            Destroy (this.gameObject);
        }
    }
}
```

Gambar 4.84 Script OpeningStage.cs

4.3.19 Script Timer

Script gambar 4.85 ini mengatur *timer* pada tiap *stage*, apabila *timer* menunjukan angka 0 maka pemain akan berpindah ke *scene game over*.

```
public class Timer : MonoBehaviour {

    public GameObject loading;
    public GUIText timerText;
    public string stage;
    public static float time;
    int highscore;
    public static float timeLeft
    {
        get
        {
            return time;
        }
        set
        {
            time = value;
        }
    }
    void Start(){
        highscore = PlayerPrefs.GetInt ("HighScore");
    }
    void Update () {
        if(LimitScore.Scored > highscore){
            highscore = LimitScore.Scored;
            PlayerPrefs.SetInt ("HighScore", highscore);
            PlayerPrefs.Save ();
        }
        timeLeft -= Time.deltaTime;
        timerText.text =(timeLeft).ToString("0");
        if (timeLeft < 0)
        {
            loading.SetActive (true);
            SceneManager.LoadScene("GameOver", LoadSceneMode.Single);
            Button.theStage = stage;
            Destroy (this.gameObject);
        }
    }
}
```

Gambar 4.85 Script Timer.cs

4.3.20 Script Limit dan Score

Script gambar 4.86 ini mengatur variabel *limit* dan *score* untuk target tantangan dan *score* yang diperoleh pada tiap *stage game*.

```
public class LimitScore
{
    private static int limit = 0;
    public static int Limit
    {
        get
        {
            return limit;
        }
        set
        {
            limit = value;
        }
    }

    private static int score = 0;
    public static int Scored
    {
        get
        {
            return score;
        }
        set
        {
            score = value;
        }
    }
}
```

Gambar 4.86 Script LimitScore.cs

4.3.21 Script HighScore

Script ini mengatur *highscore* yang terdapat pada game, dapat diamati pada gambar 4.87.

```
public class highscore : MonoBehaviour {

    // Use this for initialization
    public GUIText text;
    int hscore;
    void Start () {
        hscore = PlayerPrefs.GetInt ("HighScore");
    }

    // Update is called once per frame
    void Update () {
        text.text = "HighScore: "+hscore.ToString ();
    }
}
```

Gambar 4.87 Script highscore.cs

4.3.22 Script Teks Score

Script pada gambar 4.88 ini digunakan untuk mengatur *score* yang ditampilkan pada tiap *stage*.

```
public class scoretext : MonoBehaviour {

    // Use this for initialization
    public GUIText MyText;
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        MyText.text = "Score: "+LimitScore.Scored.ToString ();
    }
}
```

Gambar 4.88 Script scoretext.cs

4.3.23 Script Fitur Stage

Script gambar 4.89 ini mengatur fitur perpindahan *stage* dengan tombol *keyboard* setelah pemain menyelesaikan *stage* tertentu.

```
public class DontDestroy : MonoBehaviour {

    private static DontDestroy _instance;
    void Awake()
    {
        //if we don't have an [_instance] set yet
        if(!_instance)
            _instance = this ;
        //otherwise, if we do, kill this thing
        else
            Destroy(this.gameObject) ;


        DontDestroyOnLoad(this.gameObject) ;
    }


    // Update is called once per frame
    void Update () {
        if (Input.GetKeyDown(KeyCode.Alpha0))
        {
            print("Menu pressed");
            SceneManager.LoadScene("MainMenu", LoadSceneMode.Single);
        }
        if (Input.GetKeyDown(KeyCode.Alpha1)&& PlayerPrefs.GetInt ("Stage")
>=1)
        {
```

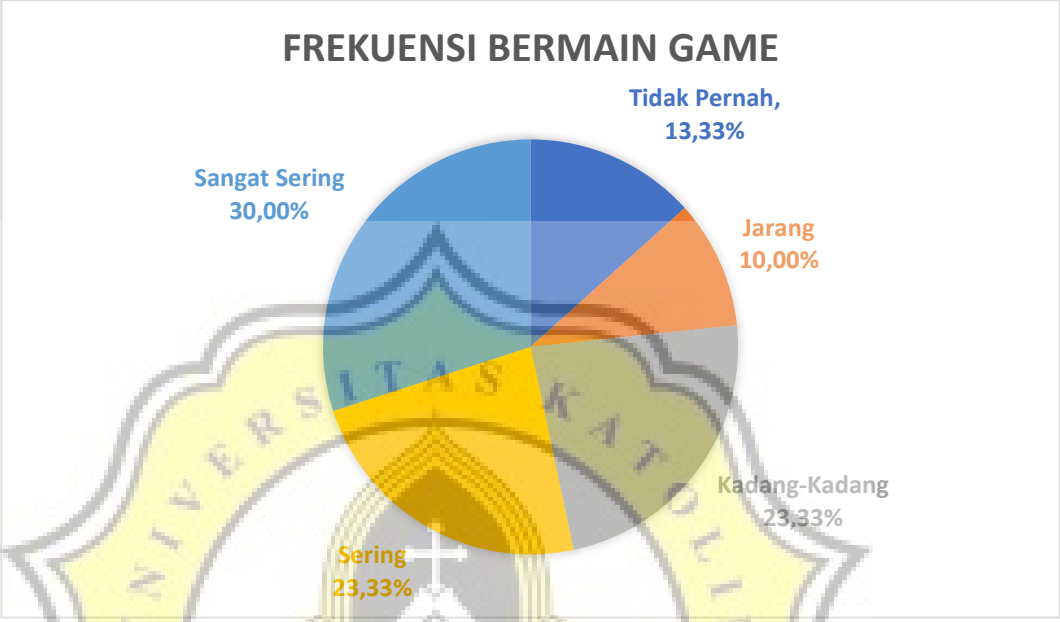
```
            print("1 pressed");
            SceneManager.LoadScene("Stage1", LoadSceneMode.Single);
        }
        if (Input.GetKeyDown(KeyCode.Alpha2)&& PlayerPrefs.GetInt ("Stage")
>=2)
        {
            print("2 pressed");
            SceneManager.LoadScene("Stage2", LoadSceneMode.Single);
        }
        if (Input.GetKeyDown(KeyCode.Alpha3)&& PlayerPrefs.GetInt ("Stage")
>=3)
        {
            print("3 pressed");
            SceneManager.LoadScene("Stage3", LoadSceneMode.Single);
        }
        if (Input.GetKeyDown(KeyCode.Alpha4)&& PlayerPrefs.GetInt ("Stage")
>=4)
        {
            print("4 pressed");
            SceneManager.LoadScene("Stage4", LoadSceneMode.Single);
        }
        if (Input.GetKeyDown(KeyCode.Alpha5)&& PlayerPrefs.GetInt ("Stage")
>=5)
        {
            print("5 pressed");
            SceneManager.LoadScene("Stage5", LoadSceneMode.Single);
        }
        if (Input.GetKeyDown(KeyCode.Alpha6)&& PlayerPrefs.GetInt ("Stage")
>=6)
        {
            print("6 pressed");
            SceneManager.LoadScene("Stage6", LoadSceneMode.Single);
        }
    }
}
```

Gambar 4.89 Script DontDestroy.cs

4.4 Analisa Data Responden

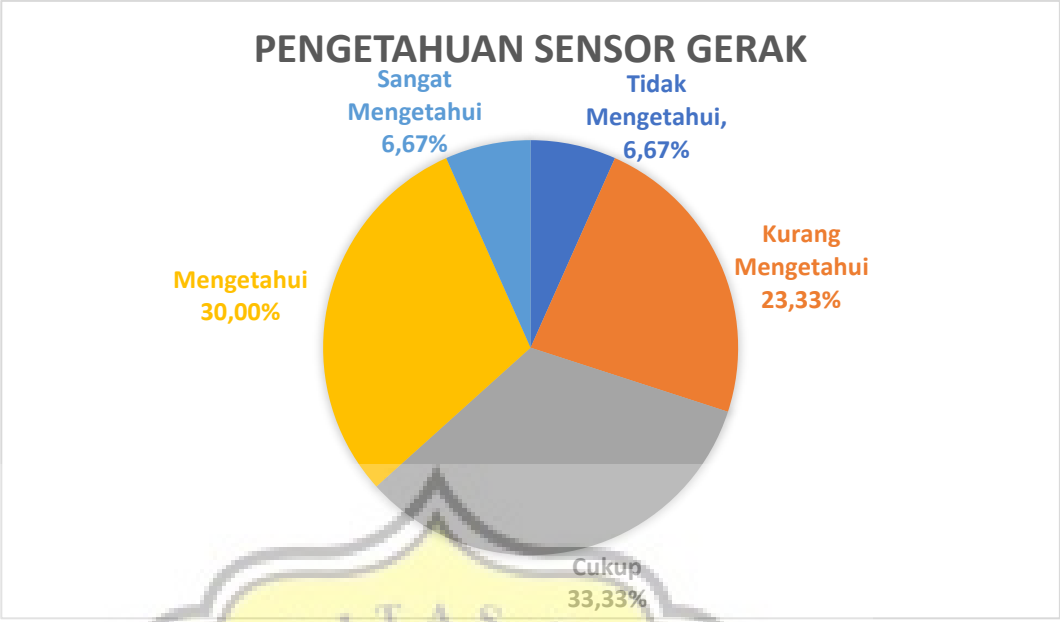
Setelah melakukan uji coba *game* "Mari Mencerna" dan melakukan pengumpulan data kuesioner *pos test* dari 30 siswa di beberapa SMP di Kota Semarang, diperoleh hasil sebagai berikut. Berdasarkan frekuensi bermain *game* dapat dilihat pada diagram 4.1, sebanyak 30% siswa menyatakan bahwa mereka sangat sering bermain *game*, lalu sebanyak 23,33% siswa menyatakan bahwa sering bermain *game*. Kemudian sebanyak 23,33% siswa menyatakan bahwa mereka kadang-kadang bermain *game*. Sementara

10% siswa menyatakan jarang dan 13,33% menyatakan tidak pernah bermain *game*. Berdasarkan hasil tersebut dapat dinyatakan bahwa mayoritas responden tergolong sering bermain *game*.



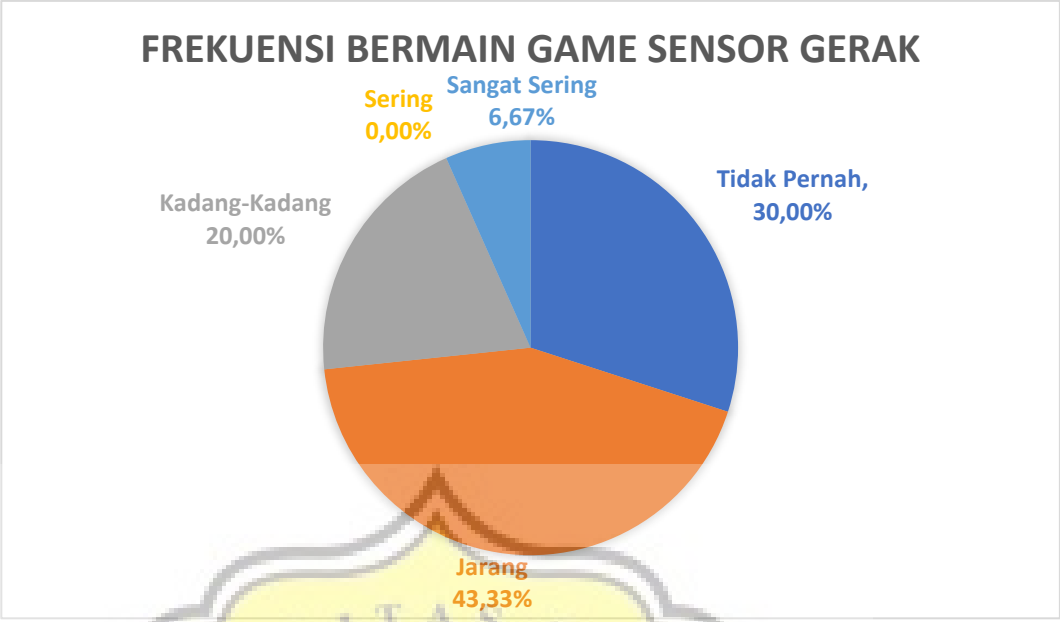
Gambar 4.90 Frekuensi Siswa Bermain Game

Selanjutnya dari pengetahuan responden mengenai sensor gerak yang dapat dilihat pada diagram 4.2. Diperoleh bahwa 6,67% sangat mengetahui dan 30% mengetahui tentang sensor gerak. Kemudian sebanyak 33,33% memiliki pengetahuan yang cukup akan sensor gerak. Sisanya sebanyak 23,33% siswa kurang mengetahui dan 6,67% tidak mengetahui mengenai sensor gerak. Hal ini menunjukkan bahwa mayoritas siswa cukup mengetahui sensor gerak.



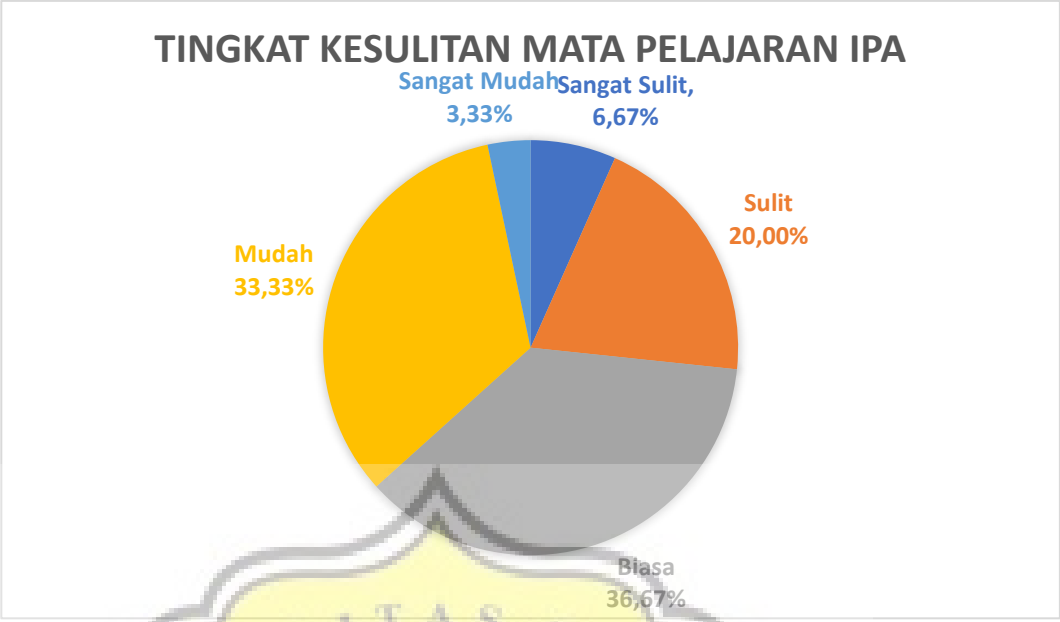
Gambar 4.91 Pengetahuan Sensor Gerak

Untuk frekuensi responden bermain *game* menggunakan sensor gerak. Sebanyak 6,67% siswa menyatakan bahwa mereka sangat sering memainkan *game* dengan sensor gerak. Sebanyak 20% siswa menyatakan kadang-kadang bermain menggunakan sensor gerak. Selanjutnya sebanyak 43,33% siswa menyatakan jarang dan sisanya 30% menyatakan tidak pernah bermain menggunakan sensor gerak. Hasil dari diagram 4.3 ini menunjukkan bahwa banyak responden belum terbiasa bermain *game* menggunakan sensor gerak.



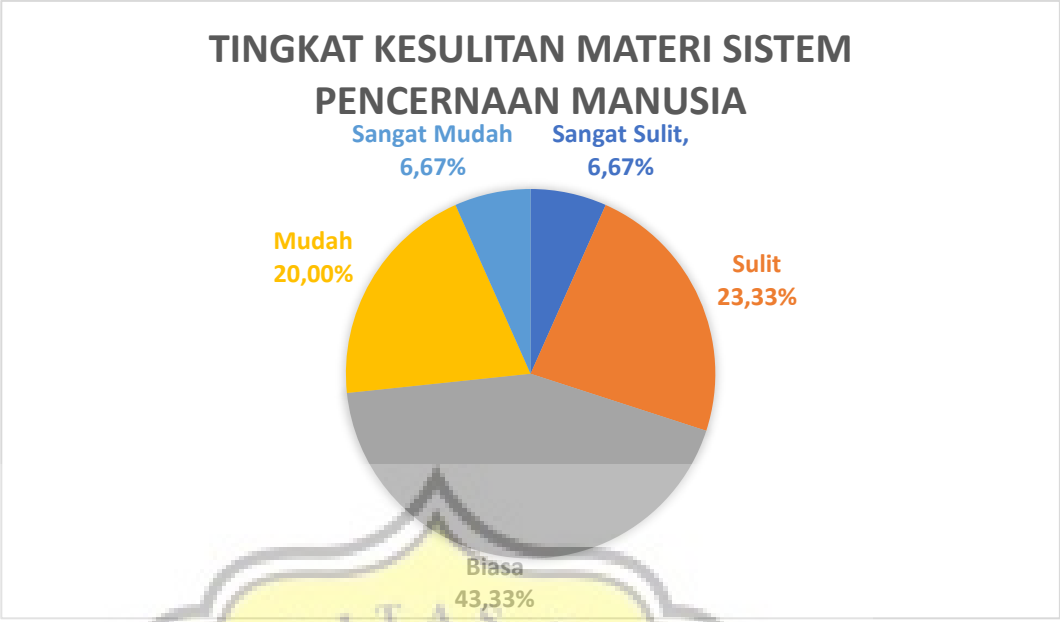
Gambar 4.92 Frekuensi Siswa Bermain Game Sensor Gerak

Berikutnya kita akan melihat pendapat responden mengenai tingkat kesulitan mata pelajaran IPA pada diagram 4.4. Responden yang merasa bahwa mata pelajaran IPA sangat mudah sebesar 3,33%, dan sebesar 33.33% merasa bahwa mudah. Sebanyak 36,67% siswa merasa tingkat kesulitan mata pelajaran IPA biasa. Dan sebanyak 20% siswa menyatakan sulit dan sisanya sebesar 6,67% menyatakan sangat sulit. Hal ini menunjukkan bahwa mayoritas siswa menganggap tingkat kesulitan mata pelajaran IPA biasa.



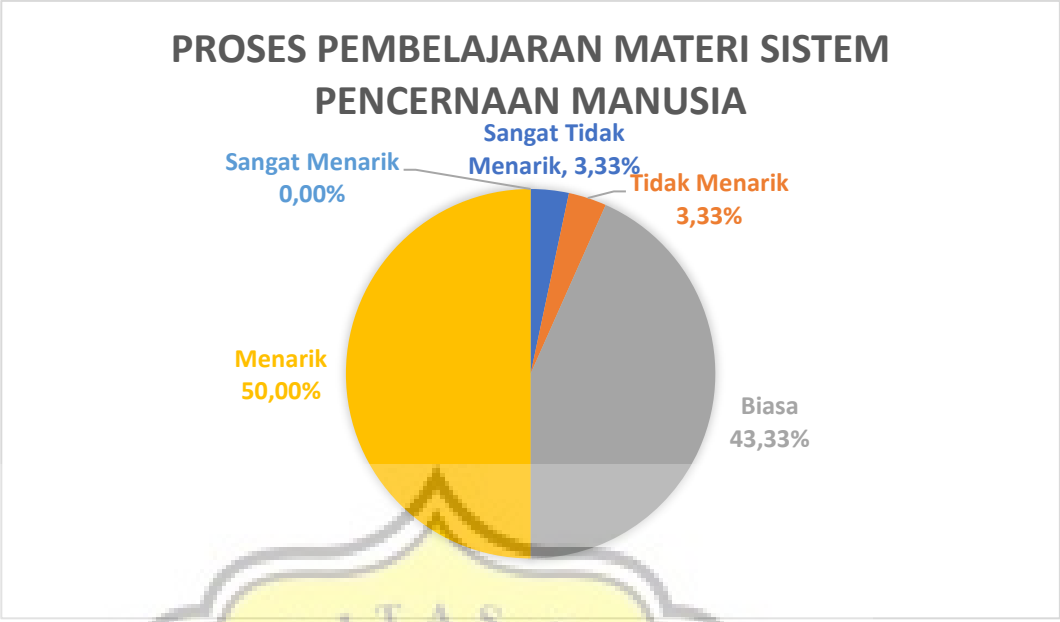
Gambar 4.93 Tingkat Kesulitan Mata Pelajaran IPA

Selanjutnya mengenai pendapat responden mengenai tingkat kesulitan materi sistem pencernaan manusia. Responden yang merasa materi tersebut sangat mudah sebesar 6,67%, dan sebesar 20% merasa bahwa mudah. Sebanyak 43,33% siswa merasa tingkat kesulitan materi sistem pencernaan manusia biasa. Dan sebanyak 23,33% siswa menyatakan sulit dan sisanya sebesar 6,67% menyatakan sangat sulit. Berdasarkan hasil diagram 4.5 ini maka mayoritas responden mengganggap bahwa materi sistem pencernaan manusia memiliki tingkat kesulitan biasa.



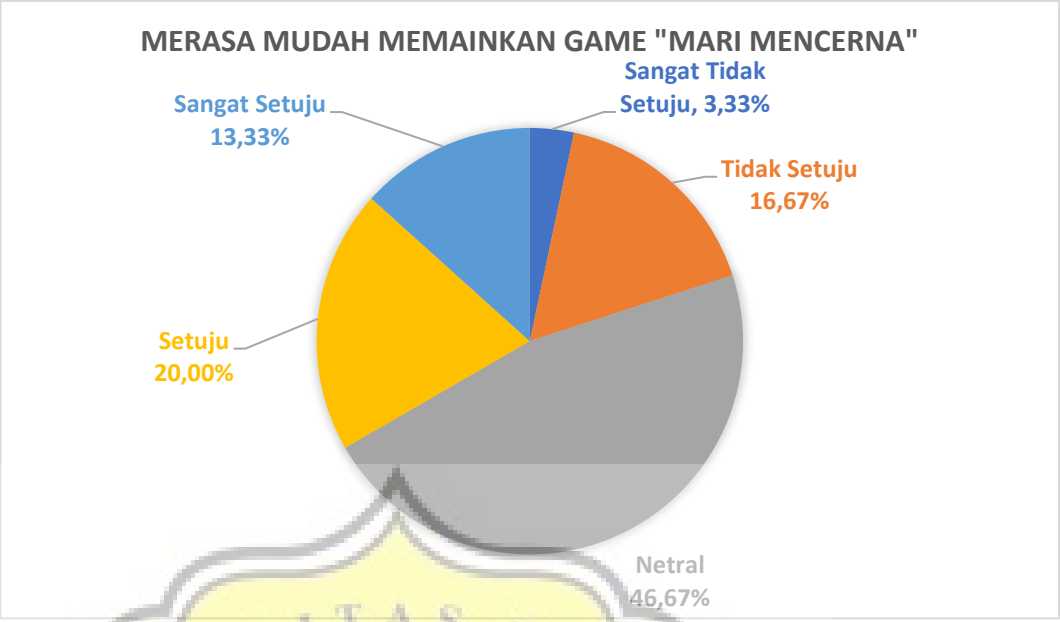
Gambar 4.94 Tingkat Kesulitan Materi Sistem Pencernaan

Dari segi pendapat responden mengenai proses pembelajaran materi sistem pencernaan manusia yang dapat dilihat pada diagram 4.6. Sebanyak 50% responden merasa proses pembelajaran telah menarik. Sedangkan sebanyak 43,33% siswa merasa bahwa proses pembelajaran biasa. Sisanya sebesar 3,33% siswa menyatakan tidak menarik dan 3,33% siswa lainnya mengganggap proses pembelajaran materi sistem pencernaan manusia sangat tidak menarik. Berdasarkan hasil tersebut mayoritas siswa telah tertarik dengan proses pembelajaran materi sistem pencernaan manusia.



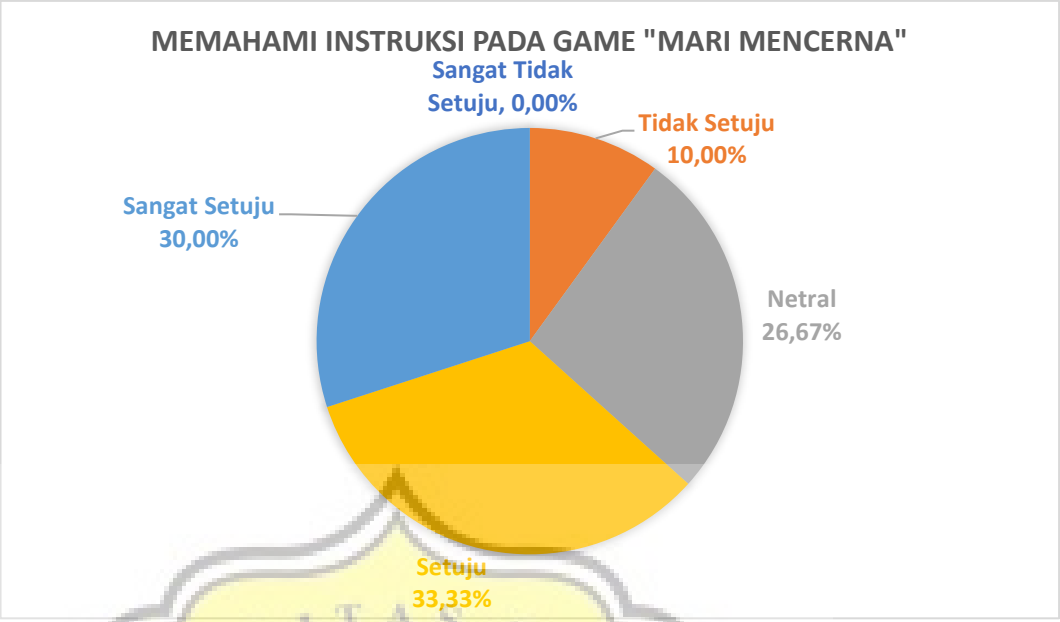
Gambar 4.95 Tingkat Ketertarikan Pembelajaran Sistem Pencernaan Manusia

Menurut pendapat siswa yang telah mencoba game "Mari Mencerna". Sebanyak 13,33% merasa sangat mudah ketika memainkan game ini, kemudian sebanyak 20% memilih setuju bahwa merasa mudah saat memainkan game ini. Sebanyak 53,85% menjawab netral. Siswa yang tidak setuju bahwa game "Mari Mencerna" mudah dimainkan sebanyak 16,67%. Dan yang merasa sangat kesulitan dalam memainkan game sebanyak 3,33%. Berdasarkan hasil pada diagram 4.7 tersebut siswa merasa cukup mudah memainkan game "Mari Mencerna", namun beberapa siswa masih merasa kesulitan dalam memainkan game ini, karena belum terbiasa memainkan game menggunakan sensor gerak.



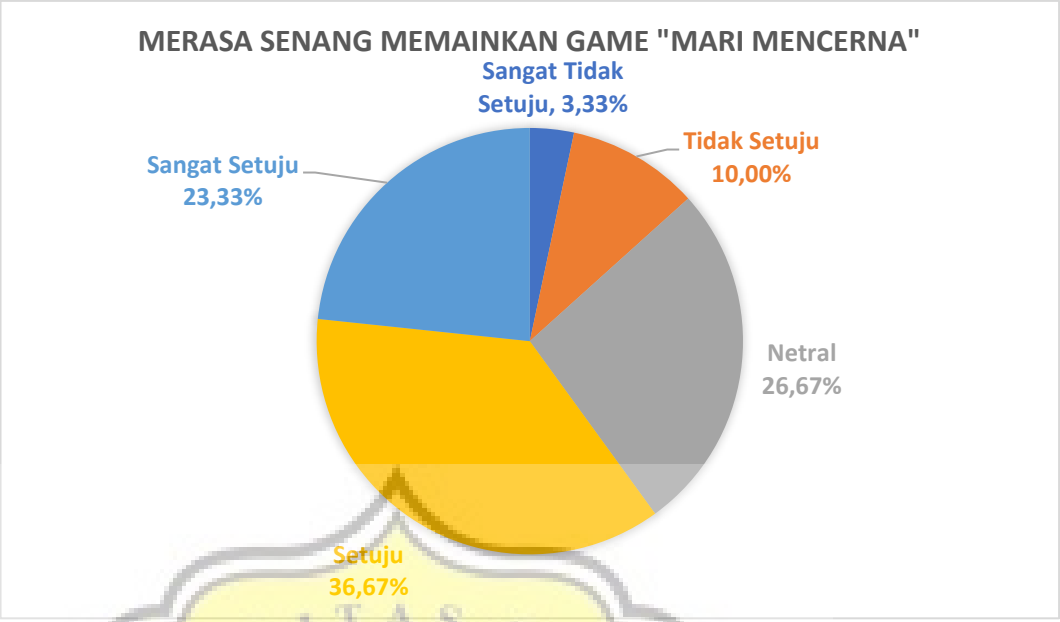
Gambar 4.96 Tingkat Kemudahan Bermain Game “Mari Mencerna”

Berdasarkan pemberian instruksi yang terdapat pada game “Mari Mencerna” seperti pada diagram 4.8. Sebanyak 30% siswa sangat setuju bahwa mereka memahami instruksi yang ada pada game. Kemudian sebanyak 33.33% siswa merasa setuju. Siswa yang memilih netral sebanyak 26,67%. Dan 10% siswa tidak setuju bahwa mereka memahami instruksi yang terdapat pada game. Dari hasil tersebut, instruksi yang diberikan pada game “Mari Mencerna” sudah dapat dipahami oleh mayoritas siswa yang mencoba game ini.



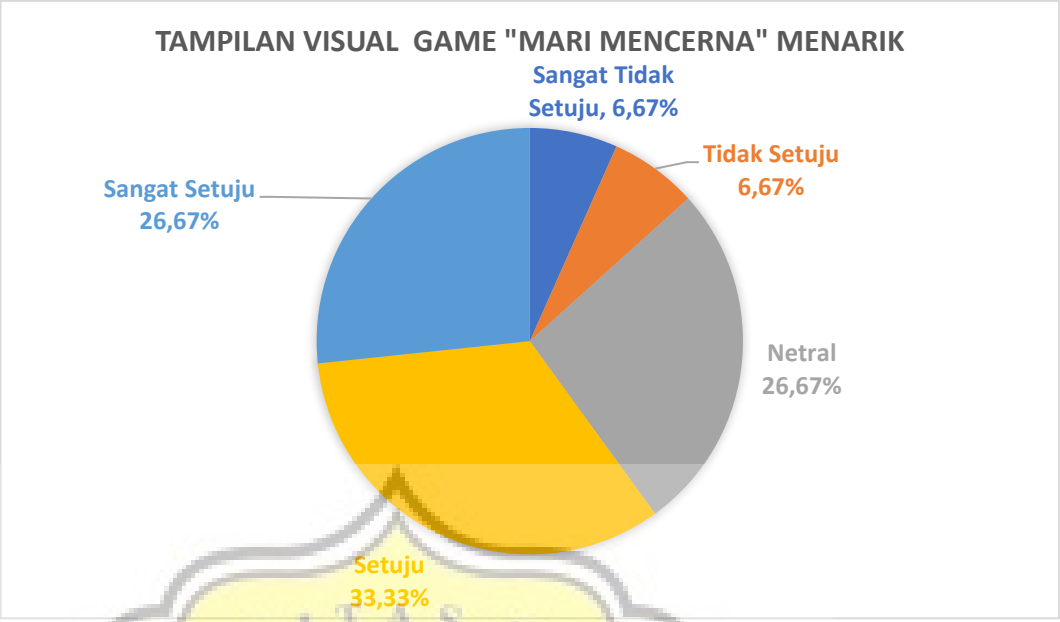
Gambar 4.97 Tingkat Pemahaman Instruksi Game "Mari Mencerna"

Dari segi rasa senang yang didapat saat memainkan game "Mari Mencerna" sebanyak 23,33% sangat setuju bahwa mereka merasa senang ketika memainkan game ini. Sedangkan 36,67% siswa merasa setuju bahwa mereka merasa senang saat bermain. Sebanyak 26,67% memilih netral. Namun ada 10% siswa yang tidak setuju & 3,33% siswa sangat tidak setuju bahwa mereka merasa senang ketika memainkan game ini. Dari hasil yang terdapat pada diagram 4.9 tersebut diperoleh bahwa game "Mari Mencerna" telah mampu memberikan perasaan senang ketika dimainkan.



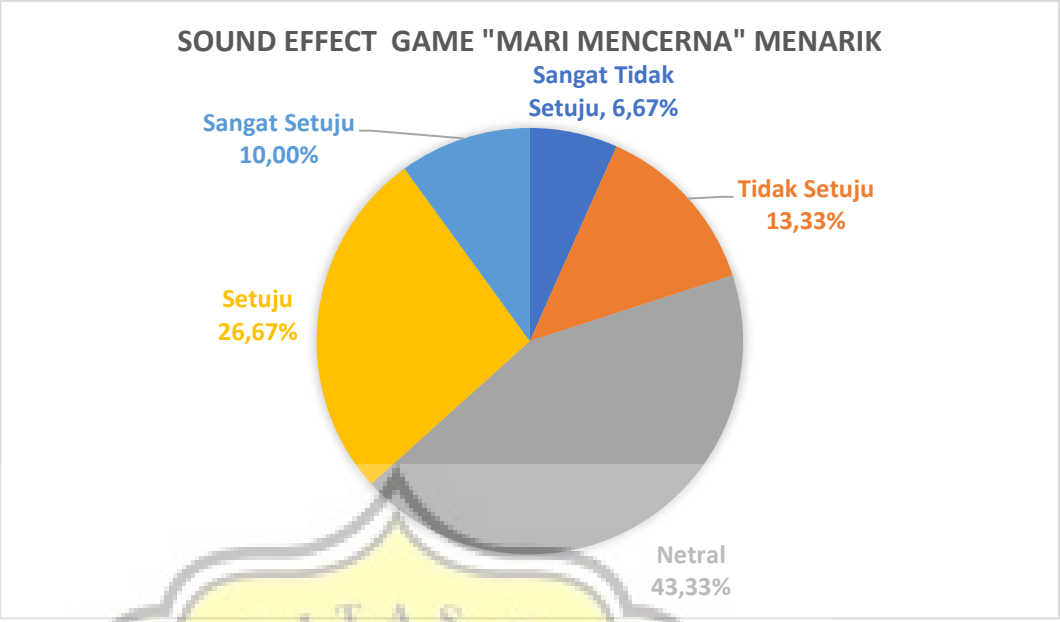
Gambar 4.98 Tingkat Kesenangan Memainkan Game "Mari Mencerna"

Berdasarkan tampilan visual game "Mari Mencerna" pada diagram 4.10 siswa yang sangat setuju bahwa tampilan visual game menarik berjumlah 26,67%. Sebanyak 33,33% setuju bahwa tampilan visual game ini menarik. Kemudian 26,67% siswa memilih netral. Selanjutnya 6,67% siswa tidak setuju dan sejumlah 6,67% pula sangat tidak setuju bahwa tampilan visual game menarik. Dari hasil tersebut dapat dinyatakan bahwa tampilan visual game "Mari Mencerna" sudah menarik.



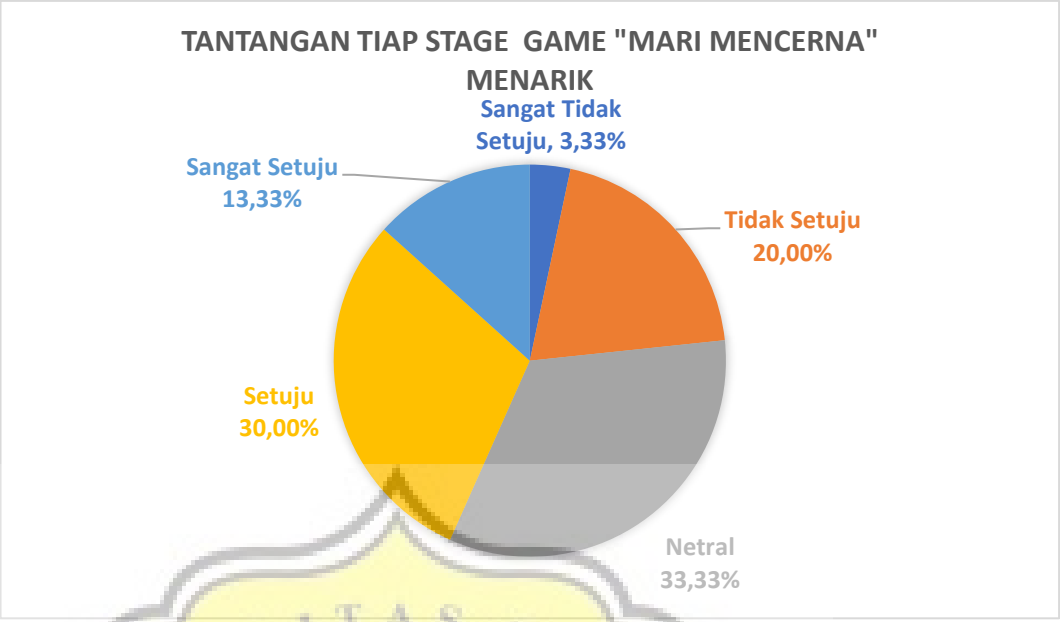
Gambar 4.99 Tingkat Ketertarikan Tampilan Visual Game "Mari Mencerna"

Berkaitan dengan *sound effect* yang digunakan pada game "Mari Mencerna", sebanyak 10% siswa sangat setuju bahwa *sound effect* pada game menarik. Kemudian sebanyak 26,67% siswa setuju dan 43,44% siswa memilih netral berkaitan dengan menariknya sound effect yang diterapkan pada game. Sebanyak 13,33% siswa tidak setuju bahwa sound effect menarik, dan 6,67% siswa menyatakan sangat tidak setuju. Berdasarkan hasil pada diagram 4.11 dapat dinyatakan bahwa *sound effect* yang diterapkan dalam game "Mari Mencerna" cukup menarik.



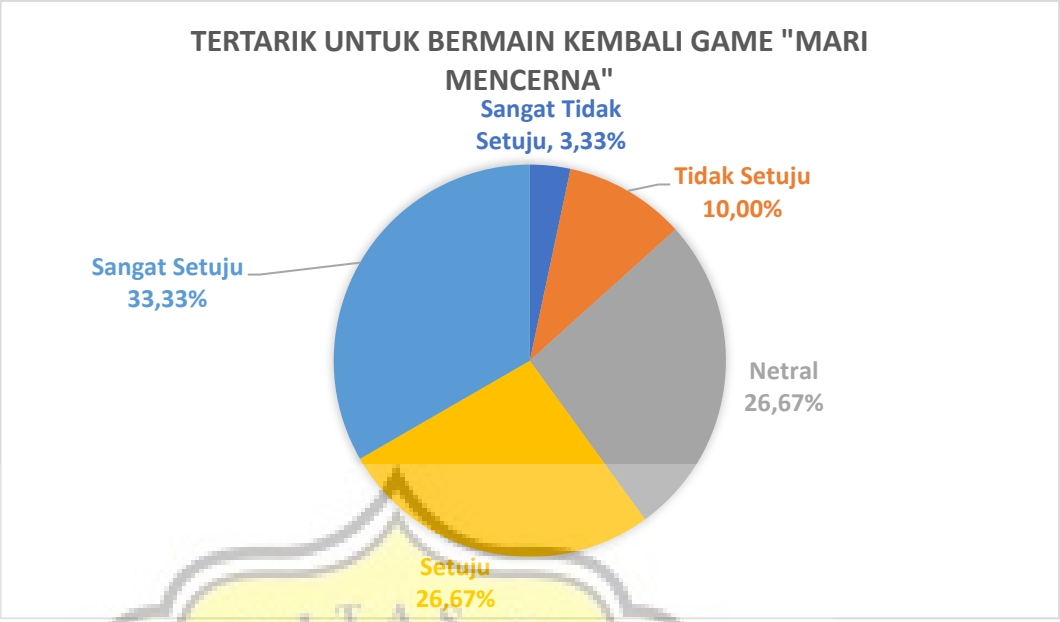
Gambar 4.100 Tingkat Ketertarikan Sound Effect Game "Mari Mencerna"

Dari segi tantangan yang diberikan pada tiap stage yang ada pada game "Mari Mencerna" seperti yang dapat dilihat pada diagram 4.12. Sebanyak 13,33% siswa sangat setuju dan 30% siswa setuju bahwa tantangan yang ada pada game menarik. Sebanyak 33,33% siswa memilih netral. Namun ada beberapa siswa yang merasa bahwa tantangan pada game "Mari Mencerna" kurang menarik bahkan tidak menarik, hal ini dapat dilihat dari pernyataan 20% siswa tidak setuju dan 3,33% sangat tidak setuju bahwa tantangan tiap stage game menarik. Berdasarkan hasil tersebut dapat dinyatakan bahwa tantangan yang diberikan pada tiap stage game "Mari Mencerna" sudah cukup menarik.



Gambar 4.101 Tingkat Ketertarikan Tantangan Game "Mari Mencerna"

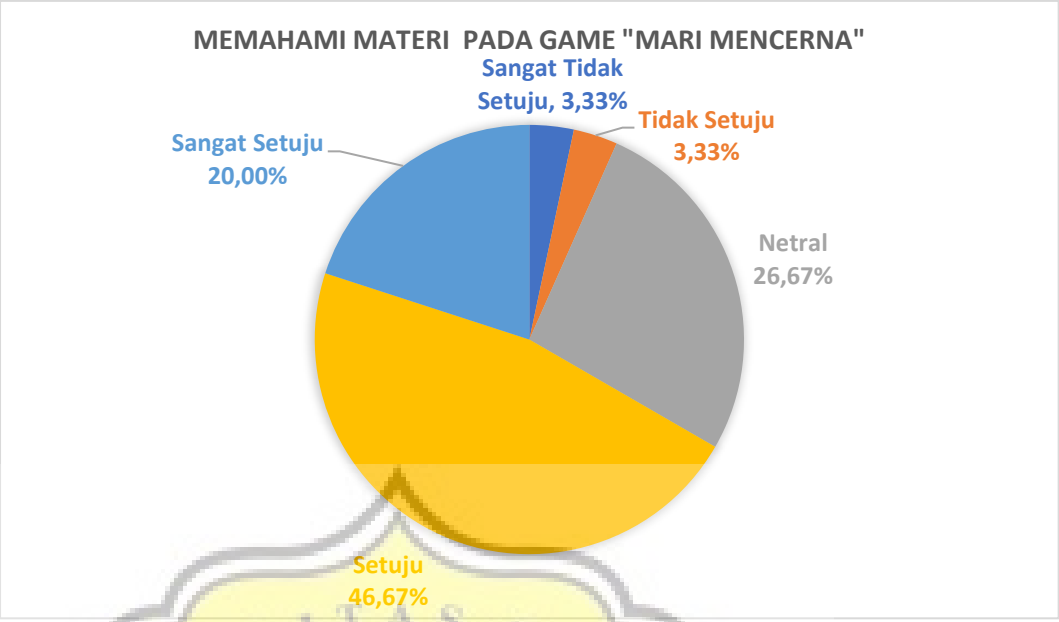
Untuk ketertarikan untuk memainkan kembali game "Mari Mencerna", sebanyak 33,33% siswa sangat setuju, dan 26,67% siswa menyatakan setuju.untuk memainkan kembali game ini. Sedangkan 26,67% siswa memilih netral. Sebanyak 10% siswa menyatakan tidak setuju dan 3,33% siswa menyatakan sangat tidak setuju untuk memainkan kembali game "Mari Mencerna". Dari hasil pada diagram 4.13 tersebut dapat disimpulkan bahwa game "Mari Mencerna" mampu membuat para siswa tertarik untuk memainkan game ini kembali.



Gambar 4.102 Tingkat Ketertarikan Memainkan Game Kembali

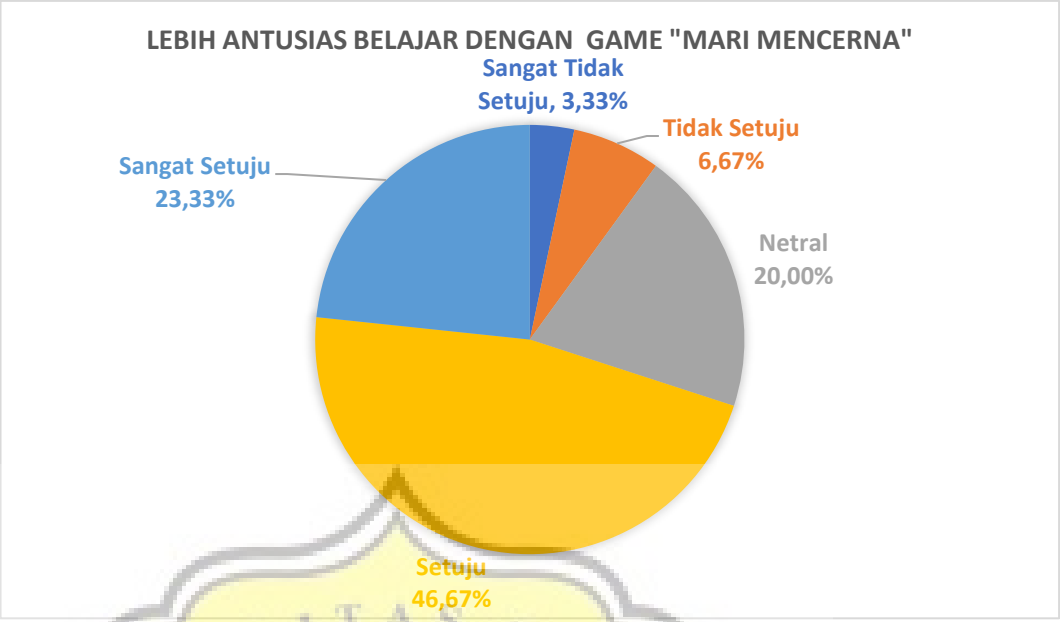
4.5 Dampak Game "Mari Mencerna"

Salah satu dampak yang diperoleh dengan memainkan *game* "Mari Mencerna" yaitu memahami materi sistem pencernaan manusia. Berdasarkan data kuesioner yang diperoleh seperti pada diagram 4.14 sebanyak 20% siswa menyatakan sangat setuju bahwa mereka memahami materi yang ada pada *game* "Mari Mencerna". Sebanyak 46,67% siswa menyatakan setuju atas pemahaman materi ini. Kemudian 26,67% siswa memilih netral, dan sisanya sebesar 3,33% siswa memilih tidak setuju dan 3,33% siswa memilih sangat tidak setuju atas pemahaman materi yang terdapat pada *game* ini. Hasil ini menunjukkan bahwa melalui *game* "Mari Mencerna" siswa dapat memahami materi sistem pencernaan manusia yang disampaikan.



Gambar 4.103 Tingkat Pemahaman Materi pada Game "Mari Mencerna"

Dampak lain yang diperoleh dari memainkan *game* "Mari Mencerna" yaitu menjadi lebih antusias untuk belajar berkaitan dengan materi sistem pencernaan manusia. Sebanyak 23,33% siswa menyatakan sangat setuju bahwa mereka menjadi lebih antusias, sedangkan 46,67% siswa menyatakan setuju. Kemudian 20% siswa menyatakan netral, dan sebanyak 6,67% tidak setuju bahwa mereka lebih antusias belajar setelah memainkan *game* ini. Sisanya sebesar 3,33% menyatakan sangat tidak setuju bahwa ia menjadi lebih antusias. Berdasarkan data hasil kuesioner pada diagram 4.15 tersebut, ternyata *game* "Mari Mencerna" mampu membuat siswa lebih antusias dalam mempelajari materi sistem pencernaan manusia.



Gambar 4.104 Tingkat Antusiasme Belajar dengan Game "Mari Mencerna"

4.6 Hasil Uji Statistik

Pada tabel 4.1 dapat dilihat tingkat validitas data responden. Setiap variabel yang memiliki kesamaan mengelompok pada satu komponen. Variabel L1, L2, dan L3 merupakan variabel yang berkaitan dengan pembelajaran, hal ini terkait dengan tingkat kesulitan mata pelajaran IPA, tingkat kesulitan materi sistem pencernaan manusia, dan tingkat ketertarikan akan pembelajaran sistem pencernaan manusia. Variabel L1, L2, dan L3 mengelompok pada komponen 2 maka dapat dinyatakan valid. Selanjutnya variabel U1, U2, dan U3 merupakan variabel yang berkaitan dengan kebergunaan, hal ini terkait dengan kemudahan saat dimainkan, instruksi yang mudah dipahami, dan rasa senang saat bermain. Variabel U1, U2, dan U3 mengelompok pada komponen 1 maka dapat dinyatakan valid. Kemudian pengelompokan variabel C1, C2, dan C3 pada komponen 1 sehingga dapat dinyatakan valid. Variabel C1, C2, dan C3 tersebut merupakan variabel

kenyamanan, terkait dengan tampilan visual game, sound effect pada game, dan tantangan yang terdapat pada game. Lalu pengelompokan variabel I1, I2, dan I3 pada komponen 1 pula sehingga dapat dinyatakan valid. Variabel I1, I2, dan I3 ini terkait dengan variabel intensi yang menyatakan pemahaman materi, antusias belajar, dan niat untuk memainkan kembali.

Component Matrix[a]

	Component		
	1	2	3
L1	,313	,762	,331
L2	,076	,742	,360
L3	,227	,661	-,087
U1	,594	-,341	,473
U2	,862	,152	-,195
U3	,820	-,041	,213
C1	,804	,020	-,023
C2	,787	-,371	,194
C3	,771	-,048	-,207
I1	,650	,065	-,641
I2	,636	,143	-,463
I3	,761	-,209	,444

Extraction Method: Principal Component Analysis.

a. 3 components extracted.

Tabel 4 1 Tabel Uji Validitas

Untuk mengetahui konsistensi dari hasil kuesioner maka dilakukan uji reliabilitas. Dari hasil uji reliabilitas pada tabel 4.2, variabel L yang terkait dengan pembelajaran memiliki nilai Cronbach's Alpha 0,690 sehingga dapat dinyatakan termasuk kategori *questionable*. Untuk variabel U yang terkait dengan kebergunaan nilai Cronbach’s Alpha sebesar 0,748 sehingga termasuk kategori *acceptable*. Kemudian dari variabel C yang terkait dengan kenyamanan nilai Cronbach’s Alpha yang didapat 0,794 sehingga termasuk

kategori *acceptable.* Pada variabel I yang terkait dengan intensitas, nilai Cronbach's Alpha sebesar 0,645 termasuk kategori *questionable*.

Variabel	**Cronbach's Alpha Based on Standardized Items**	**Keterangan**
Pembelajaran (L)	,690	Questionable
Kebergunaan (U)	,748	Acceptable
Kenyamanan (C)	,794	Acceptable
Intensi Game (I)	,645	Questionable

Tabel 4.2 Tabel Uji Reliabilitas

Pada tabel 4.3 dapat diketahui korelasi yang tercipta antar variabel. Variable L tidak memiliki korelasi dengan variable lainnya. Sementara variabel U, C, dan I saling berkolerasi. Sehingga dapat dinyatakan bahwa kebergunaan, kenyamanan, dan intensi saling terkait.

		L	U	C	I
L	Pearson Correlation	1	,203	,099	,148
U	Pearson Correlation	,203	1	,833**	,751**
C	Pearson Correlation	,099	,833**	1	,769**
I	Pearson Correlation	,148	,751**	,769**	1

**. Correlation is significant at the 0.01 level (2-tailed).

Tabel 4.3 Tabel Uji Korelasi