```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class MenuUI : MonoBehaviour {

	// ? VARIABEL TEMPAT MENYIMPAN DATA
	public RectTransform title, mainPanel, leftPanel, rightPanel;
	Button[] buttons;

	[System.Serializable]
	public class EvaluasiUI {
		public RectTransform textSkor, waktu, textTerjawab, textSoal, buttonA, buttonB, buttonC, buttonD, gambarSoal;
	}
	public EvaluasiUI evalUI;
	public bool withImage;

   void Start () {
		if (SceneManager.GetActiveScene().name == "Menu")
		{
			Text version = GameObject.Find ("Version").GetComponent<Text> ();
			version.text = "v" + Application.version;
		}
		InitOrientation ();
	}
	void Update()
	{
		if (Input.GetKeyDown (KeyCode.Escape)) {
			if (SceneManager.GetActiveScene ().name == "Tentang") {
```

```
                    SceneManager.LoadScene ("Menu");
                }
            }
        }
        public void InitOrientation (Sprite gambar=null)
        {
            int width = Screen.width;
            int height = Screen.height;

            string scene = SceneManager.GetActiveScene().name;
            if (scene == "Menu")
            {
                title.anchoredPosition = new Vector2 (0, -height *
0.08125f);
                title.sizeDelta = new Vector2 (width * 0.75f, height *
0.125f);

                mainPanel.anchoredPosition = new Vector2 (0, -height *
0.03125f);
                mainPanel.sizeDelta = new Vector2 (width * 0.375f, height
* 0.75f);
            }
            else if (scene == "Tentang")
            {
                title.anchoredPosition = new Vector2 (width *
0.18359375f, height * 0.4f);
                title.sizeDelta = new Vector2 (width * 0.4f, height * 0.09f);


                leftPanel.anchoredPosition = new Vector2 (-width *
0.2578125f, 0);
```

```
                leftPanel.sizeDelta = new Vector2 (width * 0.39f, height *
0.93f);

                rightPanel.anchoredPosition = new Vector2 (width *
0.2265625f, -height * 0.0625f);
                rightPanel.sizeDelta = new Vector2 (width / 2, height *
0.79f);
        }
        else if (scene == "Evaluasi")
        {
                if (withImage)
                {
    evalUI.textSoal.sizeDelta = new Vector2(width / 2, height * 0.347f);
    evalUI.textSoal.anchoredPosition = new Vector2(width * 0.234f, height
* 0.122f);
    evalUI.gambarSoal.gameObject.SetActive(true);
    evalUI.gambarSoal.GetComponent<Image>().sprite = gambar;
                }
                else
                {
    evalUI.textSoal.sizeDelta = new Vector2(width * 0.9375f, height *
0.347f);
    evalUI.textSoal.anchoredPosition = new Vector2(0, height * 0.122f);
    evalUI.gambarSoal.gameObject.SetActive(false);
                }
                evalUI.textSkor.sizeDelta = new Vector2(width * 0.36f,
height * 0.18f);
    evalUI.textSkor.anchoredPosition = new Vector2(width * 0.17f, -height *
0.097f);
    evalUI.textTerjawab.sizeDelta = new Vector2(width * 0.3125f, height *
0.18f);
```

```
        evalUI.textTerjawab.anchoredPosition = new Vector2(-width * 0.257f, -height * 0.097f);
        evalUI.waktu.sizeDelta = new Vector2(height * 0.18f, height * 0.18f);
        evalUI.waktu.anchoredPosition = new Vector2(0, -height * 0.097f);

        evalUI.buttonA.anchoredPosition = new Vector2(-width * 0.234375f, height * 0.104f);
        evalUI.buttonB.anchoredPosition = new Vector2(width * 0.234375f, height * 0.104f);
        evalUI.buttonC.anchoredPosition = new Vector2(-width * 0.234375f, -height * 0.104f);
        evalUI.buttonD.anchoredPosition = new Vector2(width * 0.234375f, -height * 0.104f);
                }
                else if (scene == "Post-Evaluasi")
                {
        evalUI.textSoal.sizeDelta = new Vector2(width * 0.9375f, height * 0.695f);
        evalUI.textSoal.anchoredPosition = new Vector2(0, height * 0.0695f);
        evalUI.buttonA.anchoredPosition = new Vector2(-width * 0.234375f, -height * 0.104f);
        evalUI.buttonB.anchoredPosition = new Vector2(width * 0.234375f, -height * 0.104f);
                }

                buttons = GameObject.Find("Canvas").GetComponentsInChildren<Button>();
                foreach (Button button in buttons)
                {
                        RectTransform buttonRT = button.gameObject.GetComponent<RectTransform>();
```

```
                    if (buttonRT.name.EndsWith ("Kembali"))
                    {
                        buttonRT.anchoredPosition = new Vector2 (-width
* 0.07f, -height * 0.0875f);
                        buttonRT.sizeDelta = new Vector2 (height * 0.15f,
height * 0.15f);
                    }
                    else if (buttonRT.name.StartsWith ("Button") &&
scene.Contains ("Evaluasi"))
                        buttonRT.sizeDelta = new Vector2(width * 0.43f,
height * 0.139f);
                    else
                    {
                        buttonRT.sizeDelta = new Vector2 (width / 4,
height * 0.125f);
                        if (buttonRT.name.EndsWith ("Download"))
                            buttonRT.anchoredPosition = new Vector2
(0, height * 0.139f);
                    }
                }
            }

            public void Call (string command)
            {
                if (command == "Exit")
                {
                    Application.Quit ();
                }
                else if (command == "Download")
                {
```

```
                    Application.OpenURL
("https://drive.google.com/drive/folders/1j4NpsZqQSfnyQxI62TrmpclAPJN5lvaF
?usp=sharing");
                }
                else if (command == "Back")
                {
                    if (SceneManager.GetActiveScene().name != "Menu")
                        SceneManager.LoadScene ("Menu");
                }
                else
                {
                    SceneManager.LoadScene (command);
                }
        }
}

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using Vuforia;

public class MobileUI : MonoBehaviour {

    public TrackableBehaviour[] trackableBehaviours;
  Transform[] targets;
  Text3DController[] text3D;
    Button[] buttons;

  float[] valueX, valueY, valueZ, valueSX, valueSY, valueSZ;
    float scalingSpeed, maxSize, minSize;
    float defaultParuX, defaultParuZ, defaultOthersX, defaultOthersZ;
```

```
    bool textSet = false;
    public bool RotXPlus, RotXMin, RotYMin, RotYPlus, SUp, SDown, RotZPlus,
RotZMin,
                                    inside, tracked;
    public string state;

    void Awake () {
                    buttons = GameObject.Find
("Canvas").GetComponentsInChildren<Button>();
        targets = new Transform[trackableBehaviours.Length];
                    text3D = new Text3DController[trackableBehaviours.Length];
        for (int i = 0; i < trackableBehaviours.Length; i++)
        {
           targets[i] = trackableBehaviours[i].transform.GetChild(0).transform;
                            text3D[i] =
trackableBehaviours[i].gameObject.GetComponentInChildren<Text3DController
>();
        }
        valueX = new float[trackableBehaviours.Length];
        valueY = new float[trackableBehaviours.Length];
        valueZ = new float[trackableBehaviours.Length];
        valueSX = new float[trackableBehaviours.Length];
        valueSY = new float[trackableBehaviours.Length];
        valueSZ = new float[trackableBehaviours.Length];

        defaultParuX = 0;
        defaultParuZ = 180;
        defaultOthersX = -90;
        defaultOthersZ = 0;
                    InitUI ();
           }
```

```
    void Start()
    {
        inside = false;
    for (int i = 0; i < targets.Length; i++)
    {
        valueX[i] = targets[i].eulerAngles.x;
        valueY[i] = targets[i].eulerAngles.y;
        valueZ[i] = targets[i].eulerAngles.z;
        valueSX[i] = targets[i].localScale.x;
        valueSY[i] = targets[i].localScale.y;
        valueSZ[i] = targets[i].localScale.z;
    }
        Text version =
GameObject.Find("Version").GetComponent<Text>();
        version.text = Application.version;
    }
    void InitUI ()
    {
        int width = Screen.width;
        int height = Screen.height;
        float roundButtonSize = height * 0.1875f;
        foreach (Button button in buttons) {
            RectTransform buttonTF = button.transform as
RectTransform;
            if (buttonTF.name.StartsWith ("transformer"))
            {
                buttonTF.sizeDelta = new Vector2
(roundButtonSize, roundButtonSize);
                if (buttonTF.name.EndsWith ("SUp"))
                    buttonTF.anchoredPosition = new Vector2
(width * 0.1484375f, height * 0.425f);
```

```
                                else if (buttonTF.name.EndsWith ("SDown"))
                                        buttonTF.anchoredPosition = new Vector2
(width * 0.1484375f, height / 10);
                                else if (buttonTF.name.EndsWith ("RotYMin"))
                                        buttonTF.anchoredPosition = new Vector2
(width * 0.0625f, height * 0.2625f);
                                else if (buttonTF.name.EndsWith ("RotYPlus"))
                                        buttonTF.anchoredPosition = new Vector2
(width * 0.234375f, height * 0.2625f);
                        }
                        else
                        {
                                buttonTF.sizeDelta = new Vector2 (width * 0.21f,
height * 0.21f);
                                if (buttonTF.name.EndsWith("Switch"))
                                        buttonTF.anchoredPosition = new Vector2 (-
width * 0.1171875f, height * 0.125f);
                                else if (buttonTF.name.EndsWith("Reset"))
                buttonTF.anchoredPosition = new Vector2(-width * 0.1171875f, -
height * 0.125f);
                        }
                }
        }
    bool isTrackingMarker()
    {
        bool tracked = false;
        foreach (TrackableBehaviour trackable in trackableBehaviours)
        {
            TrackableBehaviour.Status status = trackable.CurrentStatus;
            if (status == TrackableBehaviour.Status.TRACKED)
            {
```

```
            tracked = true;
            string name = trackable.gameObject.name;
                                    if (name.EndsWith("Paru-Paru") && state != "Paru-
Paru")
                                                state = "Paru-Paru";
            else if (name.EndsWith("Insang") && state != "Insang")
               state = "Insang";
            else if (name.EndsWith("Katak") && state != "Katak")
               state = "Katak";
            else if (name.EndsWith("Burung") && state != "Burung")
               state = "Burung";
            else if (name.EndsWith("Sapi") && state != "Sapi")
               state = "Sapi";
                                    else if (name.EndsWith("Kura-Kura") && state !=
"Kura-Kura")
                                                state = "Kura-Kura";
         }
      }
      return tracked;
   }
         void Update () {
               if (Input.GetKeyDown (KeyCode.Escape)) {
                                    SceneManager.LoadScene ("Menu");
               }

      tracked = isTrackingMarker();
      if (!tracked)
      {
                        if (textSet)
                        {
                                    foreach (Transform target in targets)
```

```
                                        target.gameObject.SetActive(false);
        ResetObject();
                                Start();
                                textSet = false;
                        }
    foreach (Button button in buttons)
                        {
                                if (!button.name.EndsWith ("Switch"))
                button.interactable = false;
                                else
                                        button.gameObject.SetActive(false);
                        }
}
else
{
                        if (!textSet)
                        {
                                SetText ();
        ResetObject();
                                textSet = true;
                        }
    foreach (Button button in buttons)
    {
        if (!button.name.EndsWith("Switch"))
            button.interactable = true;
                                else
                                {
                                        if (state == "Paru-Paru" || state == "Sapi")
                                                button.gameObject.SetActive(true);
                                }
    }
```

```
    }

                //// if (RotXPlus && tracked) {
                ////      valueX += Time.deltaTime * 50;
                ////      target.rotation = Quaternion.Euler (new Vector3 (valueX,
valueY, valueZ));
                //// }

                //// if (RotXMin && tracked) {
                ////      valueX -= Time.deltaTime * 50;
                ////      target.rotation = Quaternion.Euler (new Vector3 (valueX,
valueY, valueZ));
                //// }
                //// if (RotZPlus && tracked) {
                ////      valueZ += Time.deltaTime * 50;
                ////      target.rotation = Quaternion.Euler (new Vector3 (valueX,
valueY, valueZ));
                //// }

                //// if (RotZMin && tracked) {
                ////      valueZ -= Time.deltaTime * 50;
                ////      target.rotation = Quaternion.Euler (new Vector3 (valueX,
valueY, valueZ));
                //// }

                if (RotYMin && tracked) {
        for (int i = 0; i < targets.Length; i++)
        {
           valueY[i] -= Time.deltaTime * 40;
           if (state == "Paru-Paru")
```

```
                targets[i].rotation = Quaternion.Euler(new Vector3(defaultParuX,
valueY[i], defaultParuZ));
            else
                targets[i].rotation = Quaternion.Euler(new Vector3(defaultOthersX,
valueY[i], defaultOthersZ));
        }
                }

                if (RotYPlus && tracked) {
        for (int i = 0; i < targets.Length; i++)
        {
            valueY[i] += Time.deltaTime * 40;
                                if (state == "Paru-Paru")
                targets[i].rotation = Quaternion.Euler(new Vector3(defaultParuX,
valueY[i], defaultParuZ));
                                else
                targets[i].rotation = Quaternion.Euler(new Vector3(defaultOthersX,
valueY[i], defaultOthersZ));

        }
                }

                if (SUp && tracked) {
                        for (int i = 0; i < targets.Length; i++)
        {
                                if (state == "Paru-Paru")
                                {
                                        scalingSpeed = 0.1f;
                                        maxSize = 0.5f;
                                }
                                else if (state == "Insang")
```

```
                        {
    scalingSpeed = 0.05f;
                                maxSize = 0.1f;
                        }
else if (state == "Katak")
{
    scalingSpeed = 0.05f;
    maxSize = 0.1f;
}
else if (state == "Burung")
{
    scalingSpeed = 0.005f;
    maxSize = 0.005f;
}
else if (state == "Sapi")
{
    scalingSpeed = 0.001f;
    maxSize = 0.005f;
}
                        else if (state == "Kura-Kura")
                        {
                                scalingSpeed = 0.05f;
                                maxSize = 0.1f;
                        }
valueSX[i] += Time.deltaTime * scalingSpeed;
if (valueSX[i] > maxSize)
    valueSX[i] = maxSize;
valueSY[i] += Time.deltaTime * scalingSpeed;
if (valueSY[i] > maxSize)
    valueSY[i] = maxSize;
valueSZ[i] += Time.deltaTime * scalingSpeed;
```

```
            if (valueSZ[i] > maxSize)
                valueSZ[i] = maxSize;
            targets[i].localScale = new Vector3(valueSX[i], valueSY[i],
valueSZ[i]);
        }
                }

                if (SDown && tracked) {
        for (int i = 0; i < targets.Length; i++)
        {
            if (state == "Paru-Paru")
            {
                scalingSpeed = 0.1f;
                minSize = 0.2f;
            }
            else if (state == "Insang")
            {
                scalingSpeed = 0.05f;
                minSize = 0.03f;
            }
            else if (state == "Katak")
            {
                scalingSpeed = 0.05f;
                minSize = 0.03f;
            }
            else if (state == "Burung")
            {
                scalingSpeed = 0.005f;
                minSize = 0.001f;
            }
            else if (state == "Sapi")
```

```
            {
               scalingSpeed = 0.05f;
               minSize = 0.002f;
            }
                                 else if (state == "Kura-Kura")
                                 {
                                          scalingSpeed = 0.05f;
                                          minSize = 0.02f;
                                 }
            valueSX[i] -= Time.deltaTime * scalingSpeed;
            if (valueSX[i] <= minSize)
               valueSX[i] = minSize;
            valueSY[i] -= Time.deltaTime * scalingSpeed;
            if (valueSY[i] <= minSize)
               valueSY[i] = minSize;
            valueSZ[i] -= Time.deltaTime * scalingSpeed;
            if (valueSZ[i] <= minSize)
               valueSZ[i] = minSize;
            targets[i].localScale = new Vector3(valueSX[i], valueSY[i],
valueSZ[i]);
         }
                  }
         }

         // ? PERINTAH RESET OBYEK
         public void ResetObject ()
         {
      for (int i = 0; i < targets.Length; i++)
      {
                           if (state == "Paru-Paru")
                           {
```

```
                                    targets[i].rotation = Quaternion.Euler(new Vector3
(defaultParuX,0,defaultParuZ));
            targets[i].localScale = new Vector3(0.2f,0.2f,0.2f);
            valueX[i] = defaultParuX;
            valueZ[i] = defaultParuZ;
                        }
                        else if (state == "Insang")
                        {
            targets[i].rotation = Quaternion.Euler(new Vector3(defaultOthersX, 90,
defaultOthersZ));
            targets[i].localScale = new Vector3(0.04f, 0.04f, 0.04f);
            valueX[i] = defaultOthersX;
            valueZ[i] = defaultOthersZ;
                        }
        else if (state == "Katak")
                        {
            targets[i].rotation = Quaternion.Euler(new Vector3(defaultOthersX, 0,
defaultOthersZ));
            targets[i].localScale = new Vector3(0.05f, 0.05f, 0.05f);
            valueX[i] = defaultOthersX;
            valueZ[i] = defaultOthersZ;
                        }
        else if (state == "Burung")
                        {
            targets[i].rotation = Quaternion.Euler(new Vector3(defaultOthersX, 0,
defaultOthersZ));
            targets[i].localScale = new Vector3(0.002f, 0.002f, 0.002f);
            valueX[i] = defaultOthersX;
            valueZ[i] = defaultOthersZ;
                        }
        else if (state == "Sapi")
```

```
                    {
            targets[i].rotation = Quaternion.Euler(new Vector3(defaultOthersX,
180, defaultOthersZ));
            targets[i].localScale = new Vector3(0.002f, 0.002f, 0.002f);
            valueX[i] = defaultOthersX;
            valueZ[i] = defaultOthersZ;
                    }
                    else if (state == "Kura-Kura")
                    {
            targets[i].rotation = Quaternion.Euler(new Vector3(defaultOthersX,
180, defaultOthersZ));
            targets[i].localScale = new Vector3(0.05f, 0.05f, 0.05f);
            valueX[i] = defaultOthersX;
            valueZ[i] = defaultOthersZ;
                    }
          valueY[i] = targets[i].eulerAngles.y;
          valueSX[i] = targets[i].localScale.x;
          valueSY[i] = targets[i].localScale.y;
          valueSZ[i] = targets[i].localScale.z;
       }
          }
          public void DownButton (string buttonName)
          {
                if (buttonName == "RotYMin")
                      RotYMin = true;
                else if (buttonName == "RotYPlus")
                      RotYPlus = true;
                else if (buttonName == "RotXMin")
                      RotXMin = true;
                else if (buttonName == "RotXPlus")
                      RotXPlus = true;
```

```
        else if (buttonName == "SUp")
            SUp = true;
        else if (buttonName == "SDown")
            SDown = true;
        else if (buttonName == "RotZPlus")
            RotZPlus = true;
        else if (buttonName == "RotZMin")
            RotZMin = true;
    }
    public void UpButton (string buttonName)
    {
        if (buttonName == "RotYMin")
            RotYMin = false;
        else if (buttonName == "RotYPlus")
            RotYPlus = false;
        else if (buttonName == "RotXMin")
            RotXMin = false;
        else if (buttonName == "RotXPlus")
            RotXPlus = false;
        else if (buttonName == "SUp")
            SUp = false;
        else if (buttonName == "SDown")
            SDown = false;
        else if (buttonName == "RotZPlus")
            RotZPlus = false;
        else if (buttonName == "RotZMin")
            RotZMin = false;
    }
```

3D

```
    public void SetText ()
    {
```

```
        foreach (Text3DController script in text3D)
        {
            if (script.transform.parent.name == state)
            {
                if (textSet)
                    inside = !inside;
                script.SetText(tracked, inside, state);
            }
        }
                    foreach (Transform obj in targets)
                    {
                            if (obj.name == state)
                            {
                obj.gameObject.SetActive(true);
                                    Renderer[] rens =
obj.GetComponentsInChildren<Renderer>();
                                    foreach (Renderer ren in rens)
                                    {
                                            if (textSet)
                                            {
                                                    if (inside)
                                                    {
                                                            if (ren.tag == "Outside")
                                                                    ren.enabled = false;
                                                    }
                                                    else
                                                    {
                                                            if (ren.tag == "Outside")
                                                                    ren.enabled = true;
                                                    }
                                            }
```

```
                        }
                    }
                    else
                    {
                        obj.gameObject.SetActive(false);
                    }
                }
            }
        }
```

```
using System;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using UnityEngine.SceneManagement;
using UnityEngine;
using UnityEngine.UI;

public class Soal : MonoBehaviour
{
    public int soal;
    public int life = 3;
    public int score = 0;
    public float timeLimit = 60;
    int lastLife, lastScore, lastSoal;
    float lastTime;
    AudioSource player;
    public AudioClip right,wrong;
    public Text soalT,a,b,c,d,skor,timer,terjawab;
    public Sprite[] gambar;

    public MenuUI menuSC;
```

```
    void Start()
    {
        if (SceneManager.GetActiveScene().name == "Evaluasi")
        {
            player = GetComponent<AudioSource>();
            ChangeState("Read");
            if ((soal > 15 && life > 0) || life <= 0 || timeLimit <= 0)
                SceneManager.LoadScene("Post-Evaluasi");
            else
                GetSoal();
        }
        else
        {
            ChangeState("Read");
            soalT.text = "Kamu telah berhasil menyelesaikan soal evaluasi ini dengan skor terakhir:\n\n" + score;
        }
    }
    void Update ()
    {
        if (SceneManager.GetActiveScene().name == "Evaluasi")
        {
            if (soal <= 15)
            {
                timeLimit -= Time.deltaTime;
                GetParams();
            }
            if (timeLimit <= 0)
                Endgame ();
            if (Input.GetKeyDown (KeyCode.Escape))
                BackToMenu();    }
        else
```

```
    {
        if (Input.GetKeyDown(KeyCode.Escape))
            SceneManager.LoadScene("Menu");
    }
}
void Endgame ()
{
    ChangeState("Write");
    SceneManager.LoadScene("Post-Evaluasi");
}

public void BackToMenu ()
{
    ChangeState("Write");
    SceneManager.LoadScene("Menu");
}
public void Restart ()
{
    DefaultState();
    ChangeState("Write");
    SceneManager.LoadScene("Evaluasi");
}
void GetSoal ()
{
    terjawab.text = "" + (soal-1) + " / 15 terjawab";
    if (soal == 2 || soal == 3 || soal == 4)
    {
        menuSC.withImage = true;
        menuSC.InitOrientation(gambar[0]);
    }
    else if (soal == 9 || soal == 10 || soal == 11)
    {
```

```
    menuSC.withImage = true;
    menuSC.InitOrientation(gambar[1]);
}
else
{
    menuSC.withImage = false;
    menuSC.InitOrientation();
}

if (soal == 1)
{
    soalT.text = "Berikut Ini Adalah Saluran Pernapasan Manusia, yaitu...\n"+
                 "1. Bronkus 2. Hidung   3. Alveolus\n"+
                 "4. Laring  5. Trakea   6. Faring\n"+
                 "Urutan proses masuknya udara pernapasan adalah";
    a.text = "A. 2-4-6-5-1-3";
    b.text = "B. 2-4-6-1-5-3";
    c.text = "C. 2-6-4-5-1-3";
    d.text = "D. 2-3-4-6-1-5";
}
else if (soal == 2)
{
    soalT.text = "Bagian yang diberi tanda a adalah \n";
    a.text = "A. Faring";
    b.text = "B. Trakea";
    c.text = "C. Hidung";
    d.text = "D. Bronkus";
}
else if (soal == 3)
{
    soalT.text = "Bagian yang diberi tanda c adalah \n";
    a.text = "A. Trakea";
```

```
    b.text = "B. Laring";
    c.text = "C. Hidung";
    d.text = "D. Faring";
}
else if (soal == 4)
{
    soalT.text = "Bagian yang diberi tanda d adalah \n";
    a.text = "A. Trakea";
    b.text = "B. Faring";
    c.text = "C. Laring";
    d.text = "D. Hidung";
}
else if (soal == 5)
{
    soalT.text = "Berikut ini adalah sistem pernapasan manusia, yaitu \n"+
                 "1. Paru-Paru   2. Rongga Hidung    3. Rongga Mulut\n"+
                 "Urutan proses masuknya pernapasan adalah";
    a.text = "A. 1-2-3";
    b.text = "B. 2-1-3";
    c.text = "C. 3-1-2";
    d.text = "D. 3-2-1";
}
else if (soal == 6)
{
    soalT.text = "Berikut ini adalah sistem pernapasan reptil, yaitu... \n"+
                 "1. Bronkus 2. Trakea   3. Paru-Paru\n"+
                 "Urutan proses masuknya pernapasan adalah";
    a.text = "A. 1-2-3";
    b.text = "B. 3-2-1";
    c.text = "C. 2-3-1";
    d.text = "D. 2-1-3";
}
```

```
else if (soal == 7)
{
   soalT.text = "Berikut ini adalah sistem pernapasan mamalia, yaitu... \n"+
            "1. Bronkus 2. Trakea   3. Hidung   4. Alveolus\n"+
            "Urutan proses masuknya pernapasan adalah";
   a.text = "A. 1-2-3-4";
   b.text = "B. 3-2-1-4";
   c.text = "C. 2-3-4-1";
   d.text = "D. 3-1-2-4";
}
else if (soal == 8)
{
   soalT.text = "Berikut ini adalah sistem pernapasan burung, yaitu... \n" +
            "1. Paru-paru 2. Trakea   3. Hidung   4. Pundi-pundi Udara\n" +
            "Urutan proses masuknya pernapasan adalah";
   a.text = "A. 1-2-4-3";
   b.text = "B. 3-2-1-4";
   c.text = "C. 3-1-2-4";
   d.text = "D. 3-2-4-1";
}
else if (soal == 9)
{
   soalT.text = "Bagian yang diberi tanda A adalah \n";
   a.text = "A. Faring";
   b.text = "B. Paru-paru";
   c.text = "C. Pundi-pundi Udara";
   d.text = "D. Trakea";
}
else if (soal == 10)
{
   soalT.text = "Bagian yang diberi tanda B adalah \n";
   a.text = "A. Paru-paru";
```

```
    b.text = "B. Trakea";
    c.text = "C. Pundi-pundi Udara";
    d.text = "D. Rongga hidung";
}
else if (soal == 11)
{
    soalT.text = "Bagian yang diberi tanda D adalah \n";
    a.text = "A. Trakea";
    b.text = "B. Paru-paru";
    c.text = "C. Rongga hidung";
    d.text = "D. Pundi-pundi Udara";
}
else if (soal == 12)
{
    soalT.text = "Berikut yang bukan merupakan alat pernapasan burung adalah \n";
    a.text = "A. Paru-paru";
    b.text = "B. Trakea";
    c.text = "C. Faring";
    d.text = "D. Hidung";
}
else if (soal == 13)
{
    soalT.text = "Berikut yang bukan merupakan alat pernapasan mamalia adalah \n";
    a.text = "A. Pundi-pundi Udara";
    b.text = "B. Bronkus";
    c.text = "C. Trakea";
    d.text = "D. Hidung";
}
else if (soal == 14)
{
    soalT.text = "Alat pernapasan pada manusia adalah… \n";
    a.text = "A. Insang";
```

```
        b.text = "B. Ingsan";
        c.text = "C. Air";
        d.text = "D. Paru-paru";
    }
    else if (soal == 15)
    {
        soalT.text = "Alat pernapasan pada manusia adalah… kecuali \n";
        a.text = "A. Paru-paru";
        b.text = "B. Bronkus";
        c.text = "C. Pundi-pundi Udara";
        d.text = "D. Trakea";
    }
}
public void InputAnswer (string answer)
{
    if (answer == "A")
    {
        if (soal == 4 ||
            soal == 10 ||
            soal == 11 ||
            soal == 13 ||
            soal == 14)
            Answer(true);
        else
            Answer(false);
    }
    else if (answer == "B")
    {
        if (soal == 3 ||
            soal == 7 ||
            soal == 8)
            Answer(true);
```

```
        else
            Answer(false);
    }
    else if (answer == "C")
    {
        if (soal == 1 ||
            soal == 2 ||
            soal == 9 ||
            soal == 12 ||
            soal == 15)
            Answer(true);
        else
            Answer(false);
    }
    else if (answer == "D")
    {
        if (soal == 5 || soal == 6)
            Answer(true);
        else
            Answer (false);
    }
}

void GetParams ()
{
    skor.text = "Skor: "+score;
    timer.text = ""+timeLimit.ToString("0");
}
void Answer (bool correct)
{
    if (correct)
    {
```

```
        score+=7;
        if (score > 100)
            score = 100;
        Play (right);
    }
    else
    {
        score-=3;
        if (score < 0)
            score = 0;
        life-=1;
        Play (wrong);
    }
    soal++;
    GetSoal();
    ChangeState("Write");

    if ((soal > 15 && life > 0) || life <= 0)
        Endgame ();
}
void Play (AudioClip audio)
{
    player.clip = audio;
    player.Play();
}

// ? UNTUK MENYIMPAN DATA SOAL
[Serializable]
class SaveData
{
    public int soal, life, score;
    public float time;
```

```
    }

    void DefaultState()
    {
        soal = 1;
        life = 3;
        score = 0;
        timeLimit = 60;
    }

    void ChangeState (string state)
    {
        if (state == "Read")
        {
            if (File.Exists(Application.persistentDataPath + "/saveData.pcr"))
            {
                BinaryFormatter bf = new BinaryFormatter();
                FileStream file = File.Open(Application.persistentDataPath + "/saveData.pcr",
FileMode.Open);
                SaveData data = (SaveData)bf.Deserialize(file);
                file.Close();

                soal = data.soal;
                life = data.life;
                score = data.score;
                timeLimit = data.time;
                lastSoal = soal;
                lastLife = life;
                lastScore = score;
                lastTime = timeLimit;
            }
            else
```

```
            {
                DefaultState();
                ChangeState ("Write");
            }
        }
        else if (state == "Write")
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream file = File.Create(Application.persistentDataPath + "/saveData.pcr");
            SaveData data = new SaveData();

            lastSoal = soal;
            lastLife = life;
            lastScore = score;
            lastTime = timeLimit;
            data.soal = soal;
            data.life = life;
            data.score = score;
            data.time = timeLimit;

            bf.Serialize(file, data);
            file.Close();
        }
    }
}
```