## ПРИЛОЖЕНИЕ Б. ЛИСТИНГ КОДА

Листинг ButtonControl.cs

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.EventSystems;

public class ButtonControl : MonoBehaviour

{

public Sprite \_visibleIcon, \_hideIcon, \_icon;

private GameObject buttonClicked, model;

private string parentName;

private const string buPath = "Canvas/TextLog/Viewport/Content/";

private const string modelPath = "Assembly/";

public void ShowHideModel()

{

parentName = ("txt" + (EventSystem.current.currentSelectedGameObject.name).Substring(2)).Replace("Visibility", "");

buttonClicked = GameObject.Find(buPath + parentName + "/" + EventSystem.current.currentSelectedGameObject.name);

\_icon = buttonClicked.GetComponent<Image>().sprite;

model = GameObject.Find(modelPath + ((EventSystem.current.currentSelectedGameObject.name).Substring(2)).Replace("Visibility", ""));

if (\_icon.name == "visibleIcon")

{

if (model.transform.childCount > 0)

{

if (model.GetComponent<MeshRenderer>() != null)

{

MeshOff(model);

}

foreach (Transform child in model.transform)

{

MeshOff(child);

}

}

else

{

MeshOff(model);

}

buttonClicked.GetComponent<Image>().sprite = \_hideIcon;

}

else

{

if (model.transform.childCount > 0)

{

if (model.GetComponent<MeshRenderer>() != null)

{

MeshOn(model);

}

foreach (Transform child in model.transform)

{

MeshOn(child);

}

}

else

{

MeshOn(model);

}

buttonClicked.GetComponent<Image>().sprite = \_visibleIcon;

}

}

private void MeshOn(GameObject model)

{

model.GetComponent<MeshRenderer>().enabled = true;

model.GetComponent<MeshCollider>().enabled = true;

}

private void MeshOn(Transform child)

{

child.GetComponent<MeshRenderer>().enabled = true;

child.GetComponent<MeshCollider>().enabled = true;

}

private void MeshOff(GameObject model)

{

model.GetComponent<MeshRenderer>().enabled = false;

model.GetComponent<MeshCollider>().enabled = false;

}

private void MeshOff(Transform child)

{

child.GetComponent<MeshRenderer>().enabled = false;

child.GetComponent<MeshCollider>().enabled = false;

}

}

Листинг FreeCamera.cs

using UnityEngine;

using UnityEngine.EventSystems;

public class FreeCamera : MonoBehaviour

{

public Texture2D \_cursorDefault, \_cursorMove;

public float \_movementSpeed = 10f;

public float \_fasterMovementSpeed = 5f;

public float \_lookingAroundSensitivity = 3f;

private float rotX, rotY;

private bool lookingAround = false;

private Vector3 defaultPosition;

private void Start()

{

defaultPosition = gameObject.transform.position;

Cursor.SetCursor(\_cursorDefault, new Vector2(10, 5), CursorMode.ForceSoftware);

}

private void Update()

{

if (Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.UpArrow))

{

transform.position += transform.forward \* \_movementSpeed \* Time.deltaTime;

}

if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow))

{

transform.position += -transform.right \* \_movementSpeed \* Time.deltaTime;

}

if (Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.DownArrow))

{

transform.position += -transform.forward \* \_movementSpeed \* Time.deltaTime;

}

if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow))

{

transform.position += transform.right \* \_movementSpeed \* Time.deltaTime;

}

if (Input.GetKey(KeyCode.Q))

{

transform.position += transform.up \* \_movementSpeed \* Time.deltaTime;

}

if (Input.GetKey(KeyCode.E))

{

transform.position += -transform.up \* \_movementSpeed \* Time.deltaTime;

}

if (Input.GetKey(KeyCode.R))

{

transform.position = defaultPosition;

transform.localEulerAngles = Vector3.zero;

}

if (lookingAround)

{

rotX = transform.localEulerAngles.y + Input.GetAxis("Mouse X") \* \_lookingAroundSensitivity;

rotY = transform.localEulerAngles.x - Input.GetAxis("Mouse Y") \* \_lookingAroundSensitivity;

transform.localEulerAngles = new Vector3(rotY, rotX, 0f);

}

if (Input.GetAxis("Mouse ScrollWheel") != 0 && !EventSystem.current.IsPointerOverGameObject())

{

if (Input.GetAxis("Mouse ScrollWheel") < 0)

{

\_movementSpeed -= \_fasterMovementSpeed;

}

else if (Input.GetAxis("Mouse ScrollWheel") > 0)

{

\_movementSpeed += \_fasterMovementSpeed;

}

}

if (Input.GetKeyDown(KeyCode.Mouse1))

{

StartLooking();

}

else if (Input.GetKeyUp(KeyCode.Mouse1))

{

StopLooking();

}

}

private void OnDisable()

{

StopLooking();

}

private void StartLooking()

{

lookingAround = true;

Cursor.SetCursor(\_cursorMove, new Vector2(10, 5), CursorMode.ForceSoftware);

}

private void StopLooking()

{

lookingAround = false;

Cursor.SetCursor(\_cursorDefault, new Vector2(10, 5), CursorMode.ForceSoftware);

}

}

Листинг MenuPrefabDestroy.cs

using UnityEngine;

public class MenuPrefabDestroy : MonoBehaviour

{

private void OnTriggerEnter(Collider collision)

{

Destroy(collision.gameObject);

}

}

Листинг MenuPrefabMover.cs

using UnityEngine;

public class MenuPrefabMover : MonoBehaviour

{

public float \_speed = 15f;

private new Rigidbody rigidbody;

private void Start()

{

rigidbody = GetComponent<Rigidbody>();

rigidbody.velocity = new Vector3(0, \_speed, 0);

}

private void Update()

{

transform.Rotate(new Vector3(15f, 15f, 15f) \* Time.deltaTime);

}

}

Листинг MenuPrefabSpawner.cs

using UnityEngine;

using System.Collections.Generic;

public class MenuPrefabSpawner : MonoBehaviour

{

public List<GameObject> \_menuPrefabs;

public float \_minSpawnDelay = 5f;

public float \_maxSpawnDelay = 8f;

public float \_spawnXLimit = 150f;

private float random;

private GameObject prefab;

private Vector3 spawnPos;

private void Start()

{

Spawn();

}

private void Spawn()

{

random = Random.Range(-\_spawnXLimit, \_spawnXLimit);

spawnPos = new Vector3(random, -120f, 150f);

prefab = \_menuPrefabs[Random.Range(0, 47)];

prefab.transform.localScale = new Vector3(0.3f, 0.3f, 0.3f);

Instantiate(prefab, spawnPos, Quaternion.identity);

Invoke("Spawn", Random.Range(\_minSpawnDelay, \_maxSpawnDelay));

}

}

Листинг ModelClick.cs

using UnityEngine;

using UnityEngine.UI;

public class ModelClick : MonoBehaviour

{

private const string txtPath = "Canvas/TextLog/Viewport/Content/";

private Outline outline;

private GameObject txtContainer, modelToText;

private void OnMouseEnter()

{

outline = gameObject.GetComponent<Outline>();

if (outline != null)

{

outline.enabled = true;

}

else

{

gameObject.GetComponentInParent<Outline>().enabled = true;

}

}

private void OnMouseExit()

{

outline = gameObject.GetComponent<Outline>();

if (gameObject.transform.parent != null & gameObject.transform.parent.name != "Assembly")

{

txtContainer = GameObject.Find(txtPath + "txt" + gameObject.transform.parent.name);

}

else

{

txtContainer = GameObject.Find(txtPath + "txt" + gameObject.name);

}

if (outline != null & !txtContainer.GetComponent<TextClick>().\_isSelected)

{

outline.enabled = false;

}

else if (outline == null & !txtContainer.GetComponent<TextClick>().\_isSelected)

{

gameObject.GetComponentInParent<Outline>().enabled = false;

}

}

private void OnMouseDown()

{

if (gameObject.transform.parent != null & gameObject.transform.parent.name != "Assembly")

{

modelToText = GameObject.Find(txtPath + "txt" + gameObject.transform.parent.name);

SetTextComps(modelToText);

}

else

{

modelToText = GameObject.Find(txtPath + "txt" + gameObject.name);

SetTextComps(modelToText);

}

}

private void SetTextComps(GameObject model)

{

model.GetComponent<Text>().color = Color.yellow;

model.GetComponent<UnityEngine.UI.Outline>().enabled = true;

model.GetComponent<TextClick>().\_isSelected = true;

}

}

Листинг ModelRotation.cs

using UnityEngine;

public class ModelRotation : MonoBehaviour

{

public float \_rotationSpeed = 150f;

private float rotationX, rotationY;

private void OnMouseDrag()

{

rotationX = Input.GetAxis("Mouse X") \* \_rotationSpeed \* Mathf.Deg2Rad;

rotationY = Input.GetAxis("Mouse Y") \* \_rotationSpeed \* Mathf.Deg2Rad;

gameObject.transform.Rotate(rotationY, -rotationX, 0, Space.World);

}

}

Листинг ModelsMesh.cs

using UnityEngine;

using System.Linq;

public class ModelsMesh : MonoBehaviour

{

public GameObject[] models;

private Transform[] allChildren;

private MeshCollider childMesh;

private string[] noConvex = {"Hatch", "ReverseAxisBolt", "UnionStopperBolt", "TopLidThFoFork",

"TopLidLever", "OutputShaftLid", "InputShaftLid", "TopLid", "OutputShaftCoupling"};

void Start()

{

foreach (GameObject model in models)

{

if (model.transform.childCount > 0)

{

allChildren = model.GetComponentsInChildren<Transform>();

if (model.GetComponent<MeshRenderer>() != null)

{

model.AddComponent<MeshCollider>();

}

for (int i = 1; i < allChildren.Length; i++)

{

allChildren[i].gameObject.AddComponent<ModelClick>();

childMesh = allChildren[i].gameObject.AddComponent<MeshCollider>();

if (allChildren[i].name.Contains("Bolt"))

{

childMesh.convex = false;

}

else

{

childMesh.convex = true;

}

}

}

else

{

model.AddComponent<MeshCollider>();

if (noConvex.Contains(model.name))

{

model.GetComponent<MeshCollider>().convex = false;

}

else

{

model.GetComponent<MeshCollider>().convex = true;

}

}

}

}

}

Листинг MouseOver.cs

using UnityEngine;

using UnityEngine.UI;

public class MouseOver : MonoBehaviour

{

public Text textContainer;

public UnityEngine.UI.Outline textOutline;

public GameObject model;

public void OnMouseEnter()

{

textContainer.color = Color.yellow;

textOutline.enabled = true;

model.GetComponent<Outline>().enabled = true;

}

public void OnMouseExit()

{

if (!gameObject.GetComponent<TextClick>().\_isSelected)

{

textContainer.color = Color.black;

textOutline.enabled = false;

model.GetComponent<Outline>().enabled = false;

}

}

}

Листинг SceneControl.cs

using System.Collections;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class SceneControl : MonoBehaviour

{

public GameObject[] button;

public GameObject[] progressBar;

public Slider[] slider;

public Texture2D \_cursorSprite;

private GameObject currentButton, currentProgressBar;

private Slider currentSlider;

private float progress;

private void Start()

{

Cursor.SetCursor(\_cursorSprite, new Vector2(10, 5), CursorMode.ForceSoftware);

}

public void OpenMainModelScene()

{

currentButton = button[0];

currentProgressBar = progressBar[0];

currentSlider = slider[0];

StartCoroutine(LoadAsync("MainModel"));

}

public void OpenAnimationsScene()

{

currentButton = button[1];

currentProgressBar = progressBar[1];

currentSlider = slider[1];

StartCoroutine(LoadAsync("AnimationsScene"));

}

public void Exit()

{

Application.Quit();

}

public void OpenPartScene(string scenePartName)

{

SceneManager.LoadScene(scenePartName);

}

public void OpenMainModelFromPart()

{

SceneManager.LoadScene("MainModel");

}

public void OpenMainMenuScene()

{

SceneManager.LoadScene("MainMenu");

}

IEnumerator LoadAsync(string sceneName)

{

AsyncOperation operation = SceneManager.LoadSceneAsync(sceneName);

currentButton.SetActive(false);

currentProgressBar.SetActive(true);

while (!operation.isDone)

{

progress = Mathf.Clamp01(operation.progress / 0.9f);

currentSlider.value = progress;

yield return null;

}

}

}

Листинг ShowHideOnKey.cs

using UnityEngine;

using UnityEngine.UI;

public class ShowHideOnKey : MonoBehaviour

{

public GameObject assembly;

public Sprite showIcon, hideIcon;

private GameObject textContainer;

private Transform parent;

private const string buPath = "Canvas/TextLog/Viewport/Content/txt";

private void Start()

{

parent = assembly.transform;

}

private void Update()

{

if (Input.GetKeyDown(KeyCode.H))

{

ShowHide();

}

}

private void ShowHide()

{

foreach (Transform child in parent)

{

if (child.GetComponent<Outline>().enabled == true)

{

if (child.transform.childCount > 0 && child.GetComponent<MeshRenderer>() == null)

{

foreach (Transform item in child.transform)

{

MeshOff(item);

}

}

else if (child.transform.childCount > 0 && child.GetComponent<MeshRenderer>() != null)

{

foreach (Transform item in child.transform)

{

MeshOff(item);

}

MeshOff(child);

}

else

{

MeshOff(child);

}

GameObject.Find(buPath + child.name + "/bu" + child.name + "Visibility").GetComponent<Image>().sprite = hideIcon;

textContainer = GameObject.Find(buPath + child.name);

textContainer.GetComponent<UnityEngine.UI.Outline>().enabled = false;

textContainer.GetComponent<TextClick>().\_isSelected = false;

textContainer.GetComponent<Text>().color = Color.black;

}

}

}

private void MeshOff(Transform tran)

{

tran.gameObject.GetComponent<MeshRenderer>().enabled = false;

tran.gameObject.GetComponent<MeshCollider>().enabled = false;

}

}

Листинг StartAnim

using UnityEngine;

using UnityEngine.UI;

using System.Collections.Generic;

public class StartAnim : MonoBehaviour

{

public Dropdown dropdown;

public GameObject scrollBar, startButtonText;

public List<GameObject> Assemblys;

private int SceneId;

private GameObject textContainer;

private Animation currentCameraAnim, currentModelAnim, currentTextAnim, currentGasketAnim;

private string animName, modelPath;

private const string animCamera = "animCamera", animModel = "animModel", animText = "animText", gasket = "Gasket";

private const string textPath = "AnimInfoCanvas/ScrollView/Viewport/Content/animText";

private void Start()

{

dropdown.onValueChanged.AddListener(delegate { DropdownValueChanged(dropdown); });

animName = "OutputShaftCoupling";

modelPath = "Assembly/";

textContainer = GameObject.Find(textPath + animName);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

currentGasketAnim = null;

}

private void DropdownValueChanged(Dropdown change)

{

SceneId = change.value;

switch (SceneId)

{

case 0: // OutputShaftCoupling + Gasket

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop(animModel + animName + gasket);

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(0);

animName = "OutputShaftCoupling";

modelPath = "Assembly/";

currentGasketAnim = null;

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

break;

case 1: // Hatch + Gasket

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(0);

animName = "Hatch";

modelPath = "Assembly/";

currentGasketAnim = GameObject.Find(modelPath + animName + gasket).GetComponent<Animation>();

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

PrepareAnim(currentGasketAnim, animModel + animName + gasket);

break;

case 2: // InputShaftLid + Gasket

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(0);

animName = "InputShaftLid";

modelPath = "Assembly/";

currentGasketAnim = GameObject.Find(modelPath + animName + gasket).GetComponent<Animation>();

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

PrepareAnim(currentGasketAnim, animModel + animName + gasket);

break;

case 3: // CounterShaftLid + Gasket

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(0);

animName = "CounterShaftLid";

modelPath = "Assembly/";

currentGasketAnim = GameObject.Find(modelPath + animName + gasket).GetComponent<Animation>();

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

PrepareAnim(currentGasketAnim, animModel + animName + gasket);

break;

case 4: // TopLidLever

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(0);

animName = "TopLidLever";

modelPath = "Assembly/";

currentGasketAnim = null;

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

break;

case 5: // OutputShaftLid + Gasket

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(1);

animName = "OutputShaftLid";

modelPath = "Assembly1/";

currentGasketAnim = GameObject.Find(modelPath + animName + gasket).GetComponent<Animation>();

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

PrepareAnim(currentGasketAnim, animModel + animName + gasket);

break;

case 6: // TopLid + Gasket

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(1);

animName = "TopLid";

modelPath = "Assembly1/";

currentGasketAnim = GameObject.Find(modelPath + animName + gasket).GetComponent<Animation>();

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

PrepareAnim(currentGasketAnim, animModel + animName + gasket);

break;

case 7: // FirstGear

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(2);

animName = "FirstGear";

modelPath = "Assembly2/";

currentGasketAnim = null;

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

break;

case 8: // SecondGear

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(3);

animName = "SecondGear";

modelPath = "Assembly3/";

currentGasketAnim = null;

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

break;

case 9: // ThirdFourthGear

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(4);

animName = "ThirdFourthGear";

modelPath = "Assembly4/";

currentGasketAnim = null;

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

break;

case 10: // ReverseGear

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(5);

animName = "ReverseGear";

modelPath = "Assembly5/";

currentGasketAnim = null;

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

break;

case 11: // PlungerFinger

startButtonText.GetComponent<Text>().text = "Запуск анимации";

StopPreviousAnim(currentCameraAnim, animCamera + animName);

StopPreviousAnim(currentModelAnim, animModel + animName);

StopPreviousAnim(currentTextAnim, animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 0f;

currentGasketAnim[animModel + animName + gasket].time = 0f;

}

ShowHideAssembly(6);

animName = "PlungerFinger";

modelPath = "Assembly6/";

currentGasketAnim = null;

textContainer.SetActive(false);

textContainer = GameObject.Find(textPath + animName);

textContainer.SetActive(true);

currentCameraAnim = GameObject.Find("Main Camera").GetComponent<Animation>();

currentModelAnim = GameObject.Find(modelPath + animName).GetComponent<Animation>();

currentTextAnim = GameObject.Find(textPath + animName).GetComponent<Animation>();

PrepareAnim(currentCameraAnim, animCamera + animName);

PrepareAnim(currentModelAnim, animModel + animName);

PrepareAnim(currentTextAnim, animText + animName);

break;

default:

Debug.Log("Ошибка номера анимации");

break;

}

}

public void StartAnimation()

{

if (startButtonText.GetComponent<Text>().text == "Запуск анимации") {

scrollBar.GetComponent<Scrollbar>().value = 1;

currentCameraAnim.Play(animCamera + animName);

currentModelAnim.Play(animModel + animName);

currentTextAnim.Play(animText + animName);

if (currentGasketAnim != null)

{

currentGasketAnim.Play(animModel + animName + gasket);

}

currentCameraAnim[animCamera + animName].speed = 1f;

currentModelAnim[animModel + animName].speed = 1f;

currentTextAnim[animText + animName].speed = 1f;

if (currentGasketAnim != null)

{

currentGasketAnim[animModel + animName + gasket].speed = 1f;

}

startButtonText.GetComponent<Text>().text = "Перезапуск анимации";

}

else if (startButtonText.GetComponent<Text>().text == "Перезапуск анимации")

{

currentCameraAnim.Stop();

currentModelAnim.Stop();

currentTextAnim.Stop();

currentCameraAnim.Play(animCamera + animName);

currentModelAnim.Play(animModel + animName);

currentTextAnim.Play(animText + animName);

currentCameraAnim[animCamera + animName].speed = 1f;

currentModelAnim[animModel + animName].speed = 1f;

currentTextAnim[animText + animName].speed = 1f;

if (currentGasketAnim != null)

{

currentGasketAnim.Stop();

currentGasketAnim.Play(animModel + animName + gasket);

currentGasketAnim[animModel + animName + gasket].speed = 1f;

}

}

}

public void ResumeAnim()

{

currentCameraAnim[animCamera + animName].speed = 1f;

currentModelAnim[animModel + animName].speed = 1f;

currentTextAnim[animText + animName].speed = 1f;

if (currentGasketAnim != null)

{

currentGasketAnim[animModel + animName + gasket].speed = 1f;

}

}

public void PauseAnim()

{

currentCameraAnim[animCamera + animName].speed = 0f;

currentModelAnim[animModel + animName].speed = 0f;

currentTextAnim[animText + animName].speed = 0f;

if (currentGasketAnim != null)

{

currentGasketAnim[animModel + animName + gasket].speed = 0f;

}

}

public void ForwardAnim()

{

currentCameraAnim[animCamera + animName].speed += 0.5f;

currentModelAnim[animModel + animName].speed += 0.5f;

currentTextAnim[animText + animName].speed += 0.5f;

if (currentGasketAnim != null)

{

currentGasketAnim[animModel + animName + gasket].speed += 0.5f;

}

}

private void StopPreviousAnim(Animation anim, string animName)

{

anim.Stop();

anim.Play(animName);

anim[animName].speed = 0f;

anim[animName].time = 0f;

}

private void PrepareAnim(Animation anim, string animName)

{

anim.Play(animName);

anim[animName].speed = 0f;

}

private void ShowHideAssembly(int assemblyNumber)

{

for (int i = 0; i < Assemblys.Count; i++)

{

if (assemblyNumber == i)

{

Assemblys[i].transform.localScale = new Vector3(0.2f, 0.2f, 0.2f);

}

else

{

Assemblys[i].transform.localScale = Vector3.zero;

}

}

}

}

Листинг TextClick.cs

using UnityEngine;

using UnityEngine.EventSystems;

public class TextClick : MonoBehaviour, IPointerClickHandler

{

public bool \_isSelected = false;

private SceneControl \_script;

private string scenePartName;

private void Start()

{

\_script = GameObject.Find("SceneControl").GetComponent<SceneControl>();

}

public void OnPointerClick(PointerEventData pointerEventData)

{

if (pointerEventData.clickCount == 2)

{

scenePartName = gameObject.name.Substring(3);

\_script.OpenPartScene(scenePartName);

}

else

{

gameObject.GetComponent<TextClick>().\_isSelected = !gameObject.GetComponent<TextClick>().\_isSelected;

}

}

}

Листинг ToolTip.cs

using UnityEngine;

public class ToolTip : MonoBehaviour

{

public GameObject \_toolTip;

private Transform toolTipPos;

private Vector3 mousePos;

private bool IsSelected = false;

private void Start()

{

toolTipPos = \_toolTip.GetComponent<Transform>();

toolTipPos.position = new Vector3(Screen.width / 4.6f, Screen.height / 1.34f, 0);

}

private void Update()

{

if (!IsSelected)

{

mousePos = Input.mousePosition;

toolTipPos.position = new Vector3(mousePos.x + 330, mousePos.y - 252, 0);

}

}

public void ShowToolTip()

{

\_toolTip.SetActive(true);

}

public void HideToolTip()

{

if (!IsSelected)

{

\_toolTip.SetActive(false);

}

}

public void SetSelected()

{

IsSelected = !IsSelected;

}

}

Листинг ToolTipPart.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ToolTipPart : MonoBehaviour

{

public GameObject toolTip;

private Transform toolTipPos;

private Vector3 mousePos;

private bool IsSelected = false;

private void Start()

{

toolTipPos = toolTip.GetComponent<Transform>();

toolTipPos.position = new Vector3(Screen.width / 4.6f, Screen.height / 1.34f, 0);

}

private void Update()

{

if (!IsSelected)

{

mousePos = Input.mousePosition;

toolTipPos.position = new Vector3(mousePos.x + 230, mousePos.y - 252, 0);

}

}

public void ShowToolTip()

{

toolTip.SetActive(true);

}

public void HideToolTip()

{

if (!IsSelected)

{

toolTip.SetActive(false);

}

}

public void SetSelected()

{

IsSelected = !IsSelected;

}

}

Листинг Torque.cs

using UnityEngine;

public class Torque : MonoBehaviour

{

public float \_speed = 50f;

private void Update()

{

transform.Rotate(Vector3.forward \* \_speed \* Time.deltaTime);

}

}

Листинг txtOptions.cs

using UnityEngine;

using UnityEngine.UI;

public class txtOptions : MonoBehaviour

{

public Text[] textContainers;

private MouseOver mouseOver;

private new BoxCollider2D collider;

private string modelName;

private float width, height;

private void Start()

{

foreach(Text item in textContainers)

{

mouseOver = item.GetComponent<MouseOver>();

collider = item.gameObject.AddComponent<BoxCollider2D>();

mouseOver.textContainer = item;

mouseOver.textOutline = item.GetComponent<UnityEngine.UI.Outline>();

modelName = item.name.Substring(3);

mouseOver.model = GameObject.Find("Assembly/" + modelName);

height = item.rectTransform.rect.height;

width = item.rectTransform.rect.width;

collider.size = new Vector2(width, height - 5);

collider.offset = new Vector2(-0.7f, 0.5f);

}

}

}