**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ   
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Младший научный сотрудник Международной научно-учебной лаборатории интеллектуальных систем и структурного анализа, канд. Техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Максименкова  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | УТВЕРЖДАЮ  Руководитель  образовательной программы  «Программная инженерия» проф, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2020 г. |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. | RU.17701729.04.01-01 12 01-1 | | **Компьютерная игра в жанре головоломка.**  **Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.01-01 12 01-1-ЛУ**  Исполнитель:  Студентка группы БПИ 199  \_\_\_\_\_\_\_\_\_\_\_/Точилина П.В./  «\_\_\_»\_\_\_\_\_\_\_\_\_\_2020 г.    **Москва 2020** | |  |

УТВЕРЖДЕН  
RU.17701729.04.01-01 12 01-1-ЛУ

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. | RU.17701729.04.01-01 12 01-1 | | **Компьютерная игра в жанре головоломка.**  **Текст программы**  **RU.17701729.04.01-01 12 01-1**  **Листов 31** |  |

**Содержание**

[1. Введение 3](#_Toc40913340)

[1.1 Ссылка на программу: 3](#_Toc40913341)

[2. Код проекта Assembly-CSharp 4](#_Toc40913342)

[2.1 Код класса AngleBar 4](#_Toc40913343)

[2.2 Код класса AnimatorHedge 6](#_Toc40913344)

[2.3 Код класса ApplicationCloser 6](#_Toc40913345)

[2.4 Код класса CameraDrag2 7](#_Toc40913346)

[2.5 Код класса DoNotDestroy 8](#_Toc40913347)

[2.6 Код класса DragAndMove 9](#_Toc40913348)

[2.7 Код класса Globals 12](#_Toc40913349)

[2.8 Код класса MessageBoxScript 14](#_Toc40913350)

[2.9 Код класса PlayAgain 16](#_Toc40913351)

[2.10 Код класса PlayerController 16](#_Toc40913352)

[2.11 Код класса Progress 20](#_Toc40913353)

[2.12 Код класса RulesScript 22](#_Toc40913354)

[2.13 Код класса SceneLoader 24](#_Toc40913355)

[2.14 Код класса Settings 26](#_Toc40913356)

[2.15 Код класса UiController 28](#_Toc40913357)

[ЛИСТ РЕГИСТРАЦИЙ ИЗМЕНЕНИЙ 1](#_Toc40913358)

1. Введение

1.1 Ссылка на программу:

<https://github.com/LittlePotato14/TheFog>

2. Код проекта Assembly-CSharp

2.1 Код класса AngleBar

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Скрипт счётчика угла.

/// </summary>

public class AngleBar : MonoBehaviour

{

public Slider slider;

public Gradient gradient;

public Image fill;

public Text counter;

public GameObject dialogCanvas;

private Globals globals;

/// <summary>

/// Увеличивает счётчик углов на переданное значение.

/// </summary>

/// <param name="angle"> Значение увеличения. </param>

public void IncreaseAngle(float angle)

{

// Игра продолжается.

if (slider.value + angle < slider.maxValue)

{

slider.value += angle ;

fill.color = gradient.Evaluate(slider.normalizedValue);

counter.text = $"{Mathf.Round(slider.value)}/{slider.maxValue}";

}

// Уровень не пройден.

else

{

// Устанаваливаем начальные настройки.

var hero = GameObject.FindGameObjectsWithTag("hero")[0];

hero.GetComponent<Rigidbody2D>().velocity = new Vector2(0, 0);

hero.transform.rotation = Quaternion.identity;

hero.transform.Rotate(0f, 0f, globals.GetDegreeAngle(globals.startDirection));

hero.transform.position = globals.startPosition;

SetAngle(0);

// Показываем уведомление.

dialogCanvas.GetComponent<MessageBoxScript>().Show("Допустимый угол поворота был превышен.");

}

}

/// <summary>

/// Установить счётчик угла на переданное значение.

/// </summary>

/// <param name="angle"> Новое значение счётчика. </param>

public void SetAngle(float angle)

{

slider.value = angle;

fill.color = gradient.Evaluate(slider.normalizedValue);

counter.text = $"{Mathf.Round(slider.value)}/{slider.maxValue}";

}

/// <summary>

/// Вызывается один раз при запуске.

/// </summary>

void Start()

{

globals = Camera.main.GetComponent<Globals>();

slider.maxValue = globals.maxAngle;

slider.value = 0;

fill.color = gradient.Evaluate(1f);

counter.text = $"{Mathf.Round(slider.value)}/{slider.maxValue}";

}

}

2.2 Код класса AnimatorHedge

using UnityEngine;

/// <summary>

/// Скрипт анимации главного героя.

/// </summary>

public class AnimatorHedge : MonoBehaviour

{

public static Animator anim;

// Вызывается один раз при запуске.

void Start()

{

anim = GetComponent<Animator>();

}

}

2.3 Код класса ApplicationCloser

using UnityEngine;

/// <summary>

/// Скрипт закрытия приложения.

/// </summary>

public class ApplicationCloser : MonoBehaviour

{

/// <summary>

/// Закрыть.

/// </summary>

public void CloseApplication()

{

Application.Quit();

}

}

2.4 Код класса CameraDrag2

using UnityEngine;

/// <summary>

/// Скрипт передвижения камеры.

/// </summary>

public class CameraDrag2 : MonoBehaviour

{

public float dragSpeed = 10;

private Vector3 dragOrigin;

public Vector2 panLimit;

/// <summary>

/// Вызывается раз в кадр.

/// </summary>

void Update()

{

// Была нажата пкм.

if (Input.GetMouseButtonDown(1))

{

dragOrigin = Input.mousePosition;

return;

}

// Не зажата пкм.

if (!Input.GetMouseButton(1)) return;

// Двигаем камеру.

Vector3 move = Camera.main.ScreenToViewportPoint(dragOrigin - Input.mousePosition) \* dragSpeed;

transform.Translate(move, Space.World);

Vector3 pos = transform.position;

pos.x = Mathf.Clamp(pos.x, -panLimit.x, panLimit.x);

pos.y = Mathf.Clamp(pos.y, -panLimit.y, panLimit.y);

pos.z = transform.position.z;

transform.position = pos;

dragOrigin = Input.mousePosition;

}

}

2.5 Код класса DoNotDestroy

using UnityEngine;

/// <summary>

/// Для сохранения аудио между сценами.

/// </summary>

public class DoNotDestroy : MonoBehaviour

{

static GameObject instance;

void Awake()

{

if (instance != null)

{

Destroy(gameObject);

}

else

{

instance = gameObject;

DontDestroyOnLoad(gameObject);

}

}

}

2.6 Код класса DragAndMove

using UnityEngine;

/// <summary>

/// Захват и передвижение объектов.

/// </summary>

public class DragAndMove : MonoBehaviour

{

private float startPosX;

private float startPosY;

private bool selected = false;

private bool moved = false;

public GameObject sprite;

public GameObject self;

public float rotateSpeed = 150f;

public bool isBeingHeld = false;

private Globals globals;

/// <summary>

/// Вызывается один раз при запуске.

/// </summary>

void Start()

{

globals = Camera.main.GetComponent<Globals>();

}

/// <summary>

/// Вызывается раз за кадр.

/// </summary>

void Update()

{

// Если включена анимация уровня.

if (globals.Playing)

{

selected = false;

moved = false;

sprite.GetComponent<SpriteRenderer>().color = globals.nonSelected;

isBeingHeld = false;

return;

}

// Передвигается.

if (selected && isBeingHeld && !globals.Playing)

{

Vector3 mousePos;

mousePos = Input.mousePosition;

mousePos = Camera.main.ScreenToWorldPoint(mousePos);

var newPosition = new Vector3(mousePos.x - startPosX, mousePos.y - startPosY, 0);

// Если было измеение позиции.

if (gameObject.transform.localPosition != newPosition)

{

gameObject.transform.localPosition = newPosition;

moved = true;

}

}

// Поворот и удаление.

if (selected && !globals.Playing)

{

if (Input.GetKeyUp(KeyCode.Delete))

{

DestroyImmediate(self);

return;

}

float rotateVector = Input.GetAxis("Horizontal");

gameObject.transform.Rotate(0, 0, -rotateVector \* rotateSpeed \* Time.deltaTime);

}

}

/// <summary>

/// При нажатии на лкм.

/// </summary>

public void OnMouseDown()

{

if (selected && !globals.Playing && Input.GetMouseButtonDown(0))

{

isBeingHeld = true;

Vector3 mousePos;

mousePos = Input.mousePosition;

mousePos = Camera.main.ScreenToWorldPoint(mousePos);

startPosX = mousePos.x - transform.localPosition.x;

startPosY = mousePos.y - transform.localPosition.y;

}

}

/// <summary>

/// При отжатии.

/// </summary>

public void OnMouseUp()

{

// Если не был выбран объект, то теперь он выбран.

if (!selected)

{

selected = true;

sprite.GetComponent<SpriteRenderer>().color = globals.selected;

return;

}

// Если объект не был передвинут, то теперь он не выбран.

if (!moved)

{

selected = false;

sprite.GetComponent<SpriteRenderer>().color = globals.nonSelected;

}

moved = false;

isBeingHeld = false;

}

}

2.7 Код класса Globals

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Настройки уровня.

/// </summary>

public class Globals : MonoBehaviour

{

/// <summary>

/// Вызывается один раз при запуске.

/// </summary>

void Start()

{

EndRules = Progress.GetInstance().EndRule;

}

/// <summary>

/// Возвращает угол поворота по переданному вектору.

/// </summary>

/// <param name="vec"> Вектор. </param>

/// <returns> Угол в градусах. </returns>

public float GetDegreeAngle(Vector2 vec)

{

var angle = Mathf.Atan2(vec.y, vec.x) / Mathf.PI \* 180.0f;

return angle > 0 ? angle : 360 + angle;

}

/// <summary>

/// Пройдены ли правила.

/// </summary>

bool endRules = false;

public bool EndRules {

get => endRules;

set {

endRules = value;

Progress.GetInstance().EndRule = endRules;

Progress.Serialize();

}

}

public float speed = 1.5f;

public Vector2 startDirection = new Vector2(0.5f, 1);

public float maxAngle = 360;

public Vector3 startPosition = new Vector3(-3, -2, 0);

// Цвет выбранного объекта.

public Color selected = new Color(0, 255, 255);

// Цвет невыбранного объекта.

public Color nonSelected = new Color(0, 0, 0);

// Идёт ли анимация уровня.

private bool playing = false;

public bool Playing {

get => playing;

set {

playing = value;

GameObject.Find("addArrow").GetComponent<Button>().interactable = !playing;

}

}

}

2.8 Код класса MessageBoxScript

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

/// <summary>

/// Скрипт для окна сообщения/ошибки.

/// </summary>

public class MessageBoxScript : MonoBehaviour

{

public GameObject canvas;

public Text textField;

public Button playButton;

private Globals globals;

bool endLevel;

/// <summary>

/// Вызывается один раз при запуске.

/// </summary>

void Start()

{

globals = Camera.main.GetComponent<Globals>();

canvas.SetActive(false);

}

/// <summary>

/// Показать окно.

/// </summary>

/// <param name="message"> Текст сообщения. </param>

/// <param name="endLev"> Является ли окно уведомлением об окончании уровня. </param>

public void Show(string message, bool endLev = false)

{

endLevel = endLev;

textField.text = message;

canvas.SetActive(true);

playButton.interactable = false;

}

/// <summary>

/// Вызывается при нажатии на кнопку в окне.

/// </summary>

public void NotPlaying()

{

// Переход к следующему уровню.

if (endLevel)

{

var obj = Progress.GetInstance();

obj.CurrentLevel += 1;

if(obj.CurrentLevel <= obj.LevelAmount)

SceneManager.LoadScene("Map" + obj.CurrentLevel);

else

SceneManager.LoadScene("GameOver");

return;

}

globals.Playing = false;

playButton.interactable = true;

canvas.SetActive(false);

}

}

2.9 Код класса PlayAgain

using UnityEngine;

using UnityEngine.SceneManagement;

/// <summary>

/// Скрипт для повторной игры после окончания.

/// </summary>

public class PlayAgain : MonoBehaviour

{

/// <summary>

/// Нажатие на кнопку повтора.

/// Обновляем json.

/// </summary>

public void OnButton()

{

var inst = Progress.GetInstance();

inst.CurrentLevel = 1;

inst.EndRule = false;

Progress.Serialize();

SceneManager.LoadScene("Start");

}

}

2.10 Код класса PlayerController

using UnityEngine;

/// <summary>

/// Скрипт для контроллера игрока.

/// </summary>

public class PlayerController : MonoBehaviour

{

public AngleBar angleBar;

public float moveSpeed = 10f;

public float rotateSpeed = 150f;

private Globals globals;

public GameObject dialogCanvas;

/// <summary>

/// Коллизия с триггерным предметом.

/// </summary>

/// <param name="other"></param>

private void OnTriggerEnter2D(Collider2D other)

{

if (!globals.Playing)

return;

// Коллизия с предметом поворота.

if (other.gameObject.tag == "arrow")

{

float angle = other.gameObject.transform.rotation.eulerAngles.z;

var cos = Mathf.Cos(angle \* Mathf.PI / 180);

var sin = Mathf.Sin(angle \* Mathf.PI / 180);

var first = globals.GetDegreeAngle(gameObject.GetComponent<Rigidbody2D>().velocity);

var second = globals.GetDegreeAngle(new Vector2(cos, sin));

var angleDif = Mathf.Abs(first - second);

angleDif = angleDif > 180 ? 360 - angleDif : angleDif;

gameObject.GetComponent<Rigidbody2D>().velocity = new Vector2(cos, sin) \* globals.speed;

// Увеличиваем счётчик углов.

angleBar.IncreaseAngle(angleDif);

}

// Коллизия с точкой конца уровня.

else if (other.gameObject.tag == "endPoint")

{

gameObject.GetComponent<Rigidbody2D>().velocity = new Vector2(0, 0);

dialogCanvas.GetComponent<MessageBoxScript>().Show("Уровень пройден.", true);

}

// Коллизия с опасной точкой.

else if(other.gameObject.tag == "danger")

{

// Устанаваливаем начальные настройки.

SetStartSettings();

// Показываем уведомление.

dialogCanvas.GetComponent<MessageBoxScript>().Show("Вы попали в ловушку.");

}

// Коллизия с границей уровня.

else if (other.gameObject.tag == "border")

{

// Устанаваливаем начальные настройки.

SetStartSettings();

// Показываем уведомление.

dialogCanvas.GetComponent<MessageBoxScript>().Show("Вы ушли в дремучий лес.");

}

}

/// <summary>

/// Устанавливет героя в начальное положение.

/// </summary>

private void SetStartSettings()

{

gameObject.GetComponent<Rigidbody2D>().velocity = new Vector2(0, 0);

gameObject.transform.rotation = Quaternion.identity;

gameObject.transform.Rotate(0f, 0f, globals.GetDegreeAngle(globals.startDirection));

gameObject.transform.position = globals.startPosition;

angleBar.SetAngle(0);

}

/// <summary>

/// Вызывается один раз при старте.

/// </summary>

void Start()

{

globals = Camera.main.GetComponent<Globals>();

transform.rotation = Quaternion.identity;

transform.Rotate(0f, 0f, globals.GetDegreeAngle(globals.startDirection));

}

/// <summary>

/// Вызывается за фиксированное время.

/// </summary>

public void FixedUpdate()

{

// Если герой движется.

if (GetComponent<Rigidbody2D>().velocity.magnitude > 0)

{

var angle = globals.GetDegreeAngle(GetComponent<Rigidbody2D>().velocity);

transform.rotation = Quaternion.Euler(0f, 0f, angle);

AnimatorHedge.anim.SetBool("running", true);

}

// Если герой стоит.

else

{

AnimatorHedge.anim.SetBool("running", false);

}

}

}

2.11 Код класса Progress

using UnityEngine;

using System.Runtime.Serialization.Json;

using System.IO;

using System;

using System.Runtime.Serialization;

/// <summary>

/// Singleton.

/// Хранит информацию о прогрессе пользователя.

/// </summary>

[DataContract]

public class Progress

{

// Расположение файла для сериализации.

public static string path = "Progress.json";

// Единтсвенный объект класса.

private static Progress instance;

// Текущий уровень.

[DataMember]

public int CurrentLevel { get; set; } = 1;

// Количество уровней.

[DataMember]

public int LevelAmount { get; set; } = 3;

// Пройдены ли правила.

[DataMember]

public bool EndRule { get; set; }

private Progress() { }

// Реализация Singleton.

public static Progress GetInstance()

{

if (instance == null)

instance = new Progress();

return instance;

}

/// <summary>

/// Сериализует объект.

/// </summary>

public static void Serialize()

{

try

{

var formater = new DataContractJsonSerializer(typeof(Progress));

using (var fs = new FileStream(path, FileMode.Create))

formater.WriteObject(fs, GetInstance());

}

catch(Exception ex)

{

Debug.Log(ex.Message);

}

}

/// <summary>

/// Десериализует объект.

/// </summary>

public static void Deserialize()

{

try

{

var formater = new DataContractJsonSerializer(typeof(Progress));

using (var fs = new FileStream(path, FileMode.Open))

instance = (Progress)formater.ReadObject(fs);

}

catch (Exception ex)

{

Debug.Log(ex.Message);

}

}

}

2.12 Код класса RulesScript

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Скрипт для диалоговых окон с правилами игры.

/// </summary>

public class RulesScript : MonoBehaviour

{

public GameObject box;

public Text text;

// Нужен для переключения текста в окне до переключения между окнами.

private int iter = 0;

// Все предложения, которые будут показаны в окне.

public string[] sentences;

// Следующее окно.

public GameObject next;

// Является ли окно первым.

public bool isFirst = false;

private Globals globals;

/// <summary>

/// Вызывается один раз при запуске.

/// </summary>

void Start()

{

globals = Camera.main.GetComponent<Globals>();

if (!globals.EndRules && isFirst)

text.text = sentences[0];

else

box.SetActive(false);

}

/// <summary>

/// Показывает окошко с правилами.

/// </summary>

public void Show()

{

iter = 0;

box.SetActive(true);

text.text = sentences[0];

}

/// <summary>

/// Вызывается при нажатии на кнопку в окне.

/// </summary>

public void OnButton()

{

// Если текст закончился.

if (iter >= sentences.Length - 1)

{

// Открываем следующее окно.

if (next != null)

next.GetComponent<RulesScript>().Show();

// Конец правил.

else

globals.EndRules = true;

box.SetActive(false);

return;

}

// Обновление текста.

text.text = sentences[++iter];

}

}

2.13 Код класса SceneLoader

using System;

using System.IO;

using UnityEngine;

using UnityEngine.SceneManagement;

/// <summary>

/// Скрипт для загрузки сцен.

/// </summary>

public class SceneLoader : MonoBehaviour

{

/// <summary>

/// Вызывается один раз при загрузке.

/// </summary>

void Start()

{

// Если существует файл, десериализуем из него прогресс.

if (File.Exists(Progress.path))

{

Progress.Deserialize();

return;

}

// Если файла нет, сериализуем прогресс с начальными установками.

Progress.Serialize();

}

/// <summary>

/// Загрузка Сцены по имени.

/// </summary>

public void LoadScene(string name)

{

SceneManager.LoadScene(name);

}

/// <summary>

/// Нажатие на кнопку старта игры.

/// </summary>

public void OnStartButton()

{

try

{

var inst = Progress.GetInstance();

// Загрузка уровня.

if (inst.CurrentLevel <= inst.LevelAmount)

LoadScene("Map" + inst.CurrentLevel);

// Загрузка экрана окончания игры.

else

LoadScene("GameOver");

}

catch(Exception ex)

{

Debug.Log(ex.Message);

}

}

}

2.14 Код класса Settings

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Audio;

using UnityEngine.UI;

/// <summary>

/// Скрипт для сцены настроек.

/// </summary>

public class Settings : MonoBehaviour

{

public AudioMixer audioMixer;

// Массив доступных разрешений.

Resolution[] resolutions;

public Toggle isFull;

public Dropdown resDrop;

/// <summary>

/// Вызывается один раз при загрузке.

/// </summary>

void Start()

{

// Устанавливаем доступные разрешения в дропдаун.

resolutions = Screen.resolutions;

List<string> options = new List<string>();

int currentResolutionIndex = 0;

for(int i = 0; i < resolutions.Length; i++)

{

string option = resolutions[i].width + " x " + resolutions[i].height;

options.Add(option);

if (resolutions[i].Equals(Screen.currentResolution))

currentResolutionIndex = i;

}

resDrop.ClearOptions();

resDrop.AddOptions(options);

resDrop.value = currentResolutionIndex;

resDrop.RefreshShownValue();

isFull.isOn = Screen.fullScreen;

}

/// <summary>

/// Изменение громкости аудио.

/// </summary>

public void SetVolume(float volume)

{

audioMixer.SetFloat("volume", volume);

}

/// <summary>

/// Изменение галочки fullscreen.

/// </summary>

/// <param name="isFull"></param>

public void SetFullscreen(bool isFull)

{

Screen.fullScreen = isFull;

}

/// <summary>

/// Изменение разрешения.

/// </summary>

/// <param name="resIndex"></param>

public void SetResolution(int resIndex)

{

Resolution res = resolutions[resIndex];

Screen.SetResolution(res.width, res.height, Screen.fullScreen);

}

}

2.15 Код класса UiController

using UnityEngine;

using UnityEngine.SceneManagement;

/// <summary>

/// Контроллер для канваса поверх уровня.

/// </summary>

public class UiController : MonoBehaviour

{

// Счётчик углов.

public AngleBar angleBar;

// Объект для клонирования.

public Object arrow;

private Globals globals;

/// <summary>

/// Вызывается один раз при запуске.

/// </summary>

void Start()

{

globals = Camera.main.GetComponent<Globals>();

}

/// <summary>

/// Нажатие на кнопку pause.

/// </summary>

public void StartOrPause()

{

var hero = GameObject.FindGameObjectsWithTag("hero")[0];

// pause

if (globals.Playing)

{

hero.GetComponent<Rigidbody2D>().velocity = new Vector2(0, 0);

hero.transform.rotation = Quaternion.identity;

hero.transform.Rotate(0f, 0f, globals.GetDegreeAngle(globals.startDirection));

hero.transform.position = globals.startPosition;

angleBar.SetAngle(0);

}

// start

else

{

hero.GetComponent<Rigidbody2D>().AddForce(globals.startDirection \* globals.speed, ForceMode2D.Impulse);

}

globals.Playing = !globals.Playing;

}

/// <summary>

/// Добавление клона объекта поворота.

/// </summary>

public void AddArrow()

{

Instantiate(arrow, new Vector3(0, 0, 0), Quaternion.identity);

}

/// <summary>

/// Выход с уровня в стартовое меню.

/// </summary>

public void ExitLevel()

{

SceneManager.LoadScene("Start");

}

}

ЛИСТ РЕГИСТРАЦИЙ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | Новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |