Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

ЛАБОРАТОРНАЯ РАБОТА №1

По Компьютерной графике

Руководитель

Привезенцев Д.Г.
(фамилия, инициалы)

(подпись) (дата)

Студент ПИн - 121
(группа)

Ермилов М.В.
(фамилия, инициалы)

(подпись) (дата)

Муром 2023

Лабораторная работа №1

Тема: Знакомство с Unity3D. Создание простейшей игры на Unity3D

Ход работы:

1. Разработать простую 3д игру на Unity3D

.

					МИ ВлГУ 09.03.04			
Изм.	*Лист*	*№ докум.*	*Подпись*	*Дата*				
Разраб.		Ермилов М.В.				*Лит.*	*Лист*	*Листов*
Провер.		Привезенцев Д.Г.					2	13
Реценз.						ПИн-121		
Н. Контр.								
Утверд.		.						

Код проекта на юнити:

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

public class Setting
{
    static public class Control
    {
        static public class Key
        {
            static public string Jump = "space";
            static public string Left = "a";
            static public string Right = "d";
            static public string Forward = "w";
            static public string Backward = "s";
            static public string Pause = /*"e";/*/"escape";
        }
        static public class Mouse
        {
            static public float SpeedX = 3;
            static public float SpeedY = 1;
            static public bool InversionX = false;
            static public bool InversionY = false;
        }
    }
    static public class Audio
    {

        static private float _VolumeSound = 1;
        static public float VolumeSound
        {
            get => _VolumeSound;
            set
            {
                if(value >= 0 && value <= 1)
                    _VolumeSound = value;
                else if(value > 1)
                    _VolumeSound = 1;
                else if (value < 0)
                    _VolumeSound = 0;
            }
        }
        static private float _VolumeSoundCoinPickUp = 1;
        static public float VolumeSoundCoinPickUp
        {
            get => _VolumeSoundCoinPickUp;
            set
            {
                if (value >= 0 && value <= 1)
                    _VolumeSoundCoinPickUp = value;
                else if (value > 1)
                    _VolumeSoundCoinPickUp = 1;
                else if (value < 0)
                    _VolumeSoundCoinPickUp = 0;
            }
        }
        static private float _VolumeSoundGemPickUp = 1;
        static public float VolumeSoundGemPickUp
        {
            get => _VolumeSoundGemPickUp;
            set
            {
                if (value >= 0 && value <= 1)
                    _VolumeSoundGemPickUp = value;
                else if (value > 1)
                    _VolumeSoundGemPickUp = 1;
                else if (value < 0)
                    _VolumeSoundGemPickUp = 0;
            }
        }
        static private float _VolumeMusic = 0f;
        static public float VolumeMusic
        {
            get => _VolumeMusic;
            set
            {
                if (value >= 0 && value <= 1)
                    _VolumeMusic = value;
                else if (value > 1)
                    _VolumeMusic = 1;
                else if (value < 0)
                    _VolumeMusic = 0;
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Timers;
using UnityEditor;
using UnityEngine;

//[InitializeOnLoad]
public class GameInfo
{
    static public bool isBattled => TriggeredEnemy > 0;
    static public int TriggeredEnemy = 0;
    static public bool isPaused = true;
    static public int Coin = 0;
    static public int Gem = 0;
    static public float SpeedForward = 15f;
    static public float Speed = 10f;
    static public float SpeedJump = 25f;
    //static private int DayCycle = 200;
    //static private int time = 0;
    //static public int Time => time;
    // static public int DayTime => DayCycle * 2;
    //static public bool isDay => time >= 0 ? true : false;
    //static System.Timers.Timer Letimer;

    /*static GameInfo()
    {
        //Time.
        //Time.fixedDeltaTime = 0.05f;
        //Letimer = new System.Timers.Timer(100);
        //Letimer.Elapsed += timer;
        //Letimer.Start();
    }*/

    /*static void timer(object sender, ElapsedEventArgs e)
    {
        time += 1;
        if (time == DayCycle)
            time = -DayCycle;
    }*/
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using UnityEngine;

public class MonoBehaviourPaused : MonoBehaviour
{
    bool isPaused = GameInfo.isPaused;
    private void Update()
    {
        if (!GameInfo.isPaused)
            _Update();
        _UpdateNonPaused();
    }
    private void FixedUpdate()
    {
        if(isPaused != GameInfo.isPaused)
        {
            isPaused = GameInfo.isPaused;
            if(isPaused)
                _Pause();
            else
                _Play();
        }
        if (!GameInfo.isPaused)
            _FixedUpdate();
        _FixedUpdateNonPaused();
    }
    protected virtual void _Update() { }
    protected virtual void _FixedUpdate() { }
    protected virtual void _UpdateNonPaused() { }
    protected virtual void _FixedUpdateNonPaused() { }
    protected virtual void _Play() { }
    protected virtual void _Pause() { }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using UnityEngine;

public class RigidbodyGame : MonoBehaviourPaused
{
    [SerializeField]
    float DeadY = -5f;
    [SerializeField]
    bool Respawn = false;
    [SerializeField]
    Vector3 RespawnPosition = Vector3.zero;
    [SerializeField]
    bool Rotation = false;


    Rigidbody R;
    Transform T;
    private Vector3 _pausedVelocity;
    private Vector3 _pausedAngularVelocity;
    private void Start()
    {
        T = GetComponent<Transform>();
        R = GetComponent<Rigidbody>();
        R.freezeRotation = !Rotation;
    }
    protected override void _FixedUpdate()
    {
        if (T.position.y < DeadY)
        {
            if (Respawn)
                T.position = new Vector3(RespawnPosition.x, RespawnPosition.y, 
RespawnPosition.z);
            else
            {
                Debug.Log(T.position);
                Debug.Log(R.velocity);
                Destroy(gameObject);
            }
        }
    }
    protected override void _Pause()
    {
        _pausedVelocity = R.velocity;
        _pausedAngularVelocity = R.angularVelocity;
        R.isKinematic = true;
    }
    protected override void _Play()
    {
        R.isKinematic = false;
        R.velocity = _pausedVelocity;
        R.angularVelocity = _pausedAngularVelocity;
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerControl : MonoBehaviourPaused
{
    Rigidbody R;
    Transform T;

    float Speed = 12.5f;
    int isGround = 0;
    bool isJump = false;


    private void Start()
    {
        R = GetComponent<Rigidbody>();
        T = GetComponent<Transform>();
    }
    protected override void _FixedUpdate()
    {
        Move();
        RotateCamera();
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Ground")
            isGround++;
    }
    private void OnTriggerExit(Collider other)
    {
        if (other.tag == "Ground")
        {
            isGround--;
            if (isGround < 0)
                isGround = 0;
        }
    }
    private void Move()
    {
        float MoveZ = 0;
        float MoveX = 0;
        float MoveJump = 0;
        if(R.velocity.y <= 0 && isGround > 0)
            isJump = false;
        if(isGround > 0 && !isJump)
        {
            if (Input.GetKey(Setting.Control.Key.Right))
                MoveX += GameInfo.Speed;
            if (Input.GetKey(Setting.Control.Key.Left))
                MoveX -= GameInfo.Speed;
            if (Input.GetKey(Setting.Control.Key.Backward))
                MoveZ -= GameInfo.Speed;
            if (Input.GetKey(Setting.Control.Key.Forward))
                if (MoveZ != 0)
                    MoveZ = 0;
                else if(MoveX != 0)
                    MoveZ += GameInfo.Speed;
                else
                    MoveZ += GameInfo.SpeedForward;
            if (Input.GetKey(Setting.Control.Key.Jump))
            {
                MoveJump = GameInfo.SpeedJump * GameInfo.Speed;
                if(MoveX > 0)
                    MoveX += GameInfo.SpeedJump;
                if (MoveX < 0)
                    MoveX -= GameInfo.SpeedJump;
                if (MoveZ > 0)
                    MoveZ += GameInfo.SpeedJump;
                if (MoveZ < 0)
                    MoveZ -= GameInfo.SpeedJump;
                isJump = true;
            }
        }
        R.AddForce(Quaternion.Euler(0, GetComponent<Transform>().eulerAngles.y, 0) *
new Vector3(MoveX, MoveJump, MoveZ));
    }
    private void RotateCamera()
    {
        if(!Cursor.visible)
        {
            T.RotateAround(T.position, T.up,
                Input.GetAxis("Mouse X") * Setting.Control.Mouse.SpeedX *
(Setting.Control.Mouse.InversionX ? -1 : 1));
            /*T.RotateAround(T.position, T.right,
                Input.GetAxis("Mouse Y") * Setting.Control.Mouse.SpeedY *
(Setting.Control.Mouse.InversionY ? -1 : 1));*/
        }
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Rotation : MonoBehaviourPaused
{
    [SerializeField]
    float RotateInSeconds = 5.0f;
    Vector3 rotor = new Vector3(0, 0, 0);
    private void Start()
    {
        rotor = new Vector3(0, (360 / RotateInSeconds) / (1 / Time.fixedDeltaTime), 0);
    }
    protected override void _FixedUpdate()
    {
        transform.eulerAngles += rotor;
    }
}

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

public class Drop : MonoBehaviour
{
    [SerializeField]
    DropType DropType = DropType.Coin;
    [SerializeField]
    int Count = 1;

    AudioSource source;
    private void Start()
    {
        source= GetComponent<AudioSource>();
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Player")
        {
            float volume = Setting.Audio.VolumeSound;
            switch(DropType)
            {
                case DropType.Coin:
                    volume *= Setting.Audio.VolumeSoundCoinPickUp;
                    GameInfo.Coin += Count;
                    break;
                case DropType.Gem:
                    volume *= Setting.Audio.VolumeSoundGemPickUp;
                    GameInfo.Gem += Count;
                    break;
            }
            Count = 0;
            AudioSource.PlayClipAtPoint(source.clip, GetComponent<Transform>().position,
volume);
            Destroy(gameObject);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TMPro;
//using UnityEditor.PackageManager.UI;
using UnityEngine;
using UnityEngine.UI;

class PlayerUI : MonoBehaviourPaused
{
    [SerializeField]
    TextMeshProUGUI TextGem;
    [SerializeField]
    TextMeshProUGUI TextCoin;
    [SerializeField]
    GameObject MenuPaused;
    [SerializeField]
    GameObject MenuPausedSetting;

    private void Start()
    {
        UpdateText();
    }

    protected override void _FixedUpdateNonPaused()
    {
        UpdateText();
        if(Input.GetKeyDown(Setting.Control.Key.Pause))
        {
            Pause();
        }
        if (Input.GetMouseButtonDown(0) && GameInfo.isPaused)
            Play();
    }


    void Play()
    {
        Cursor.lockState = CursorLockMode.Locked;
        GameInfo.isPaused = false;
        Cursor.visible = false;
    }
    void Pause()
    {
        Cursor.lockState = CursorLockMode.None;
        GameInfo.isPaused = true;
        Cursor.visible = true;
    }
    void UpdateText()
    {
        TextGem.text = $"Gem {GameInfo.Gem}";
        TextCoin.text = $"Coin {GameInfo.Coin}";
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Music : MonoBehaviour
{

    AudioSource A;
    [SerializeField]
    AudioClip Background;
    [SerializeField]
    AudioClip Battle;

    void Start()
    {
        A = GetComponent<AudioSource>();
        A.clip = Background;
        A.loop = true;
        FixedUpdate();
    }
    void FixedUpdate()
    {

        //if (GameInfo.isBattled)
            //newAudio(Battle, true);
        //else if (!GameInfo.isBattled)
            //newAudio(Background, false);

        if (GameInfo.isPaused && A.isPlaying)
        {
            A.Pause();
        }
        else if (!GameInfo.isPaused && !A.isPlaying)
        {
            A.volume = Setting.Audio.VolumeMusic;
            A.Play();
        }

    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using UnityEngine;
using UnityEngine.AI;

public class AI_1 : MonoBehaviourPaused
{
    [SerializeField]
    AI_Trigger AreaVisibility;
    [SerializeField]
    AI_Trigger AreaPursuit;
    [SerializeField]
    float Speed = 1f;
    bool Action = false;
    NavMeshAgent N;
    Transform T;
    private void Start()
    {
        N = GetComponent<NavMeshAgent>();
        T = GetComponent<Transform>();
        N.speed = Speed;
    }
    protected override void _FixedUpdate()
    {
        if(AreaVisibility.isPlayer)
        {
            Action = true;
            GameInfo.TriggeredEnemy++;
        }
        if(!AreaPursuit.isPlayer)
        {
            Action = false;
            GameInfo.TriggeredEnemy--;
        }
        if(Action)
        {
            N.destination = AreaPursuit.Player.transform.position;
            T.LookAt(AreaPursuit.Player.transform.position);
            T.rotation = Quaternion.Euler(0, T.eulerAngles.y, 0);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using UnityEngine;

public class AI_Trigger : MonoBehaviour
{
    public GameObject Player;
    public bool isPlayer = false;
    private void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Player")
        {
            isPlayer = true;
            Player = other.gameObject;
        }
    }
    private void OnTriggerExit(Collider other)
    {
        if (other.tag == "Player")
        {
            isPlayer = false;
        }
    }
}
```

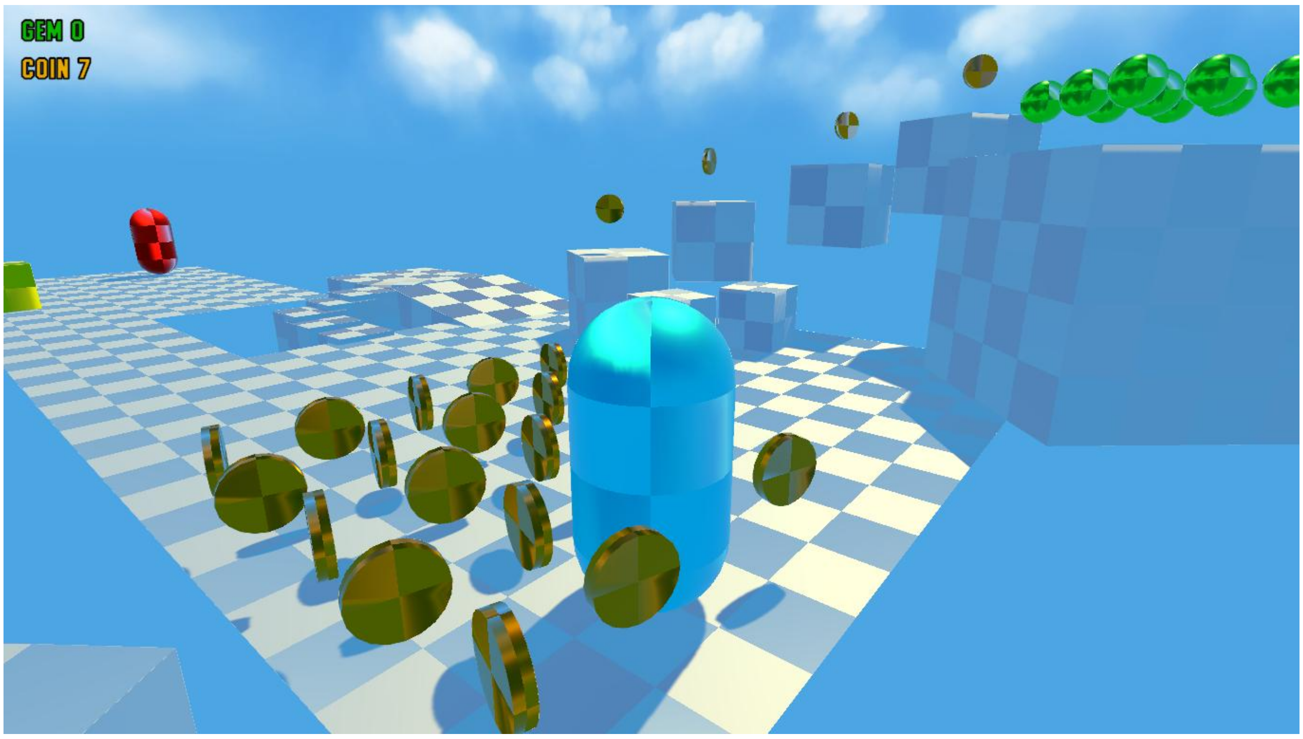

Рис 1 - пример работы игры

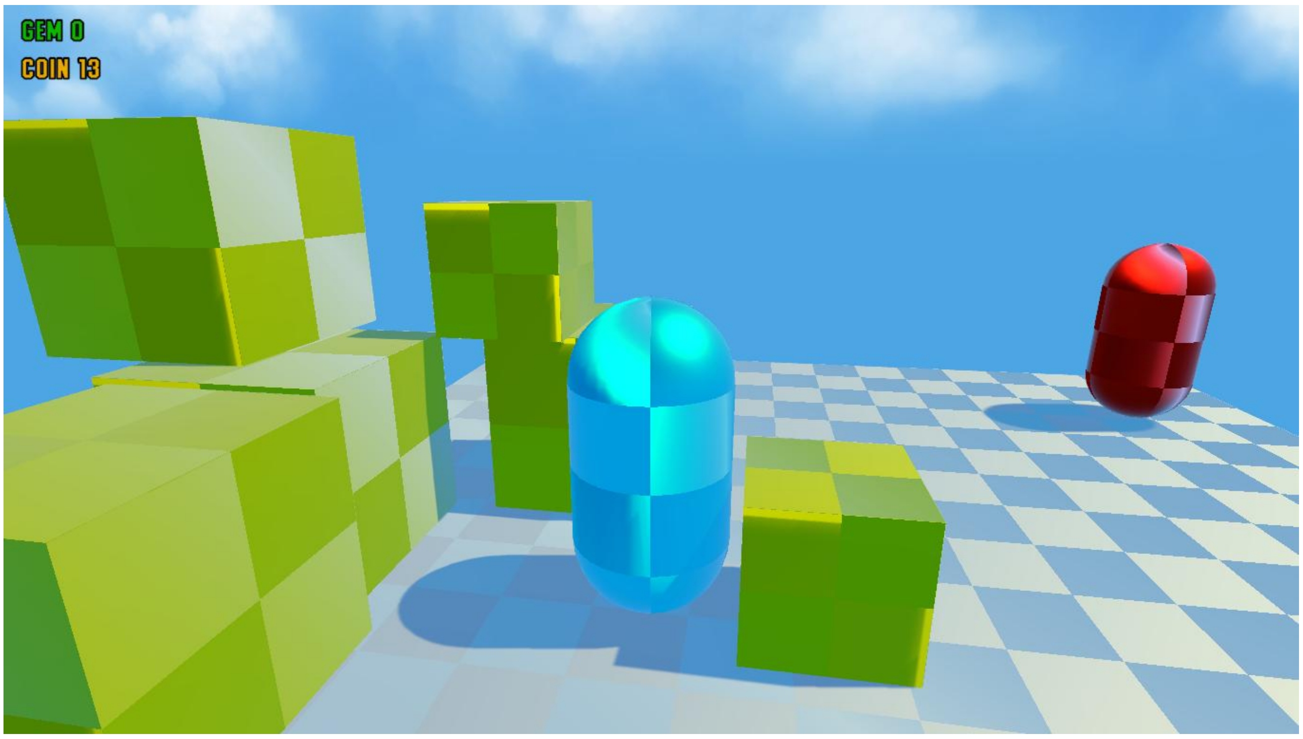

Рис 2 - пример работы игры

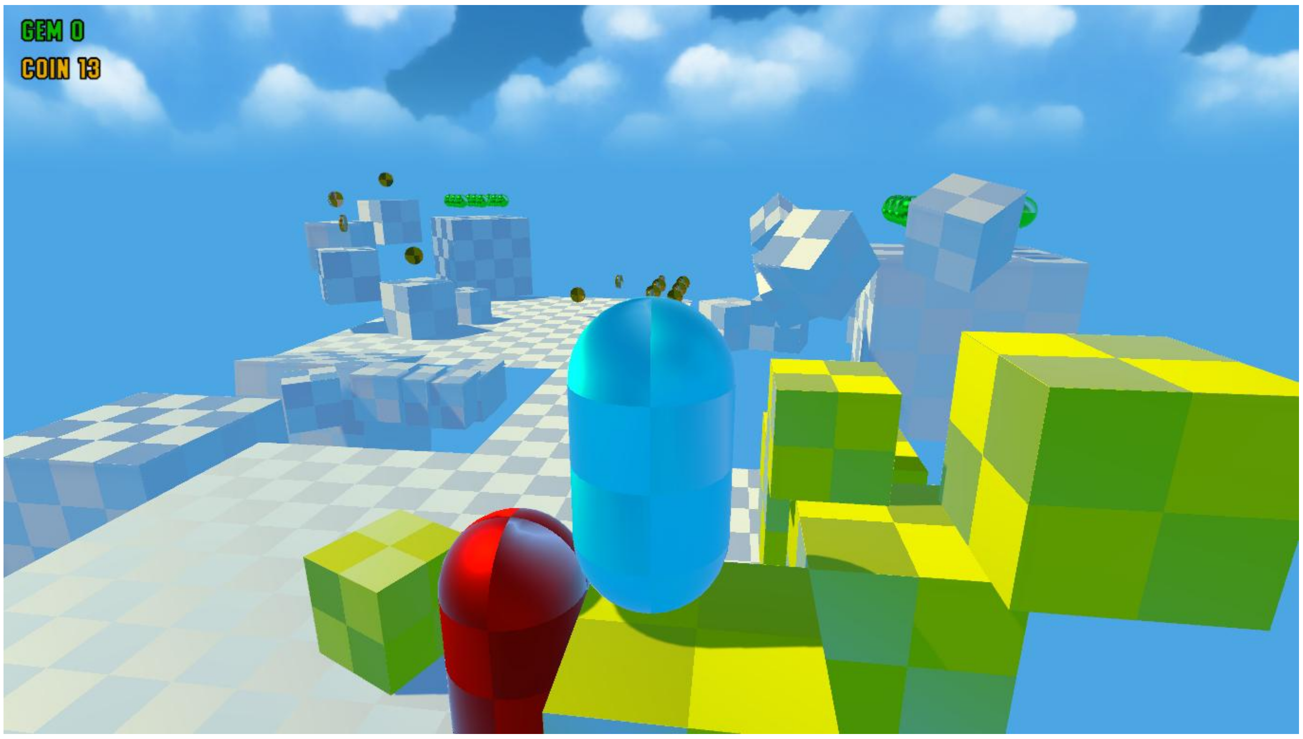

Рис 3 - пример работы игры

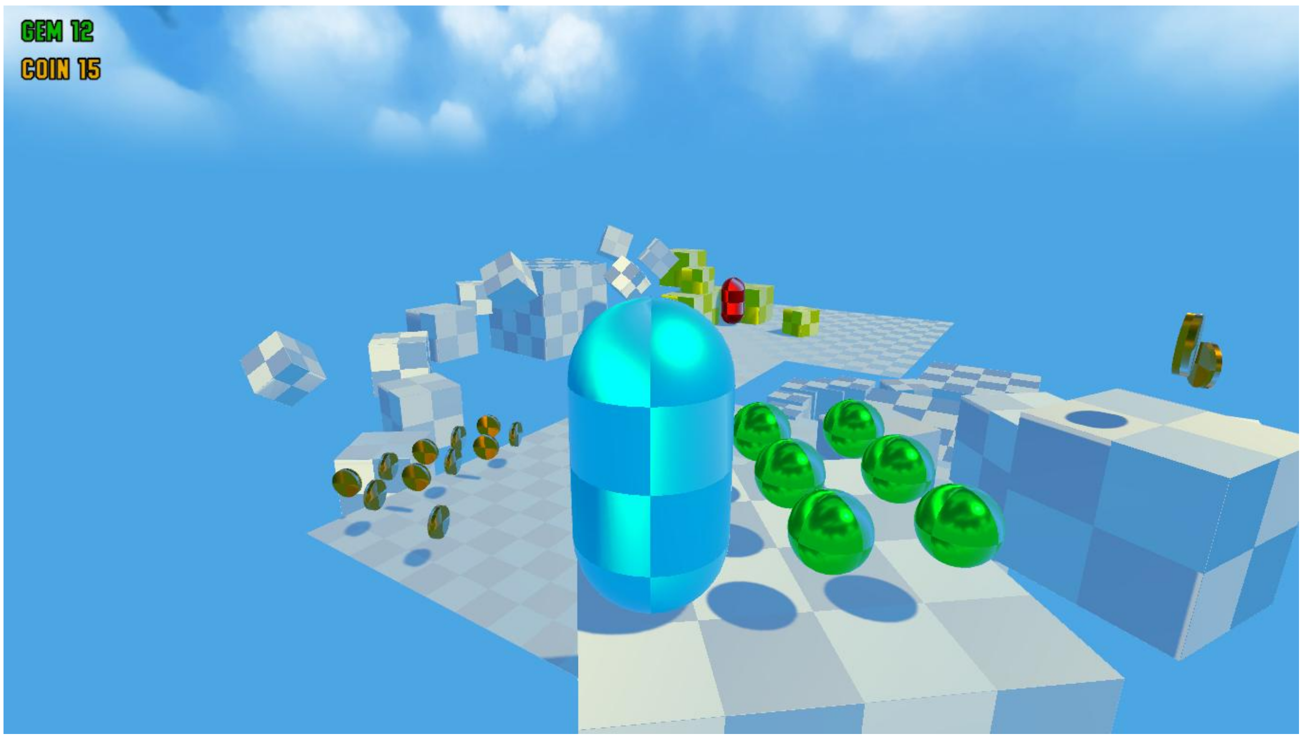

Рис 4 - пример работы игры