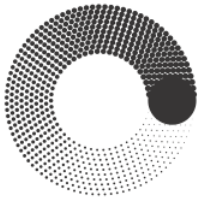
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ

Институт Принтмедиа и информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 14

Дисциплина: Компьютерная графика

Тема: Вычислительный шейдер

Выполнил: Вышегородских Виктор Егорович, **студент группы:** 211-728

_____
(Фамилия И.О.)

Дата, подпись _____ _____
(Дата) (Подпись)

Проверил: _____
_____
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____
(Дата) (Подпись)

Замечания:

_____

_____

Москва

2023

На основе теоретического материала реализовали Вычислительный шейдер. Написали необходимый скрипт. Настроили сцену.

Исходный код вычислительного шейдера:

```
#pragma kernel Spheres

#include "Random.cginc"

//variables
RWStructuredBuffer<float3> Result;
uniform float Time;

[numthreads(64,1,1)]
void Spheres (uint3 id : SV_DispatchThreadID)
{
    //generate 2 orthogonal vectors
    float3 baseDir = normalize(rand1dTo3d(id.x) - 0.5) *
(rand1dTo1d(id.x)*0.9+0.1);
    float3 orthogonal = normalize(cross(baseDir, rand1dTo3d(id.x + 7.1393) -
0.5)) * (rand1dTo1d(id.x+3.7443)*0.9+0.1);
    //scale the time and give it a random offset
    float scaledTime = Time * 2 + rand1dTo1d(id.x) * 712.131234;
    //calculate a vector based on vectors
    float3 dir = baseDir * sin(scaledTime) + orthogonal * cos(scaledTime);
    Result[id.x] = dir * 20;
}
```

Исходный код скрипта:

```
using UnityEngine;

public class BasicComputeSpheres : MonoBehaviour
{
    public int SphereAmount = 17;
    public ComputeShader Shader;

    public GameObject Prefab;

    ComputeBuffer resultBuffer;
    int kernel;
    uint threadGroupSize;
    Vector3[] output;

    Transform[] instances;

    void Start()
    {
        //program we're executing
        kernel = Shader.FindKernel("Spheres");
```

```
        Shader.GetKernelThreadGroupSizes(kernel, out threadGroupSize, out _, out
_);

        //buffer on the gpu in the ram
        resultBuffer = new ComputeBuffer(SphereAmount, sizeof(float) * 3);
        output = new Vector3[SphereAmount];

        //spheres we use for visualisation
        instances = new Transform[SphereAmount];
        for (int i = 0; i < SphereAmount; i++)
        {
            instances[i] = Instantiate(Prefab, transform).transform;
        }
    }

    void Update()
    {
        Shader.SetFloat("Time", Time.time);
        Shader.SetBuffer(kernel, "Result", resultBuffer);
        int threadGroups = (int)((SphereAmount + (threadGroupSize - 1)) /
threadGroupSize);
        Shader.Dispatch(kernel, threadGroups, 1, 1);
        resultBuffer.GetData(output);

        for (int i = 0; i < instances.Length; i++)
            instances[i].localPosition = output[i];
    }

    void OnDestroy()
    {
        resultBuffer.Dispose();
    }
}
```

Скриншоты итоговой сцены в игровом движке Unity3D: