УДК 004

Реализация диска Ньютона в Unity 3D

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается и описывается создание диска Ньютона в программе Unity. Практическим результатом будет являться диск из 7 цветов радуги и при вращении будет образовывать белый цвет.
Ключевые слова: Диск, Ньютон, Радуга

Implementation of Newton's disk in unity 3D

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses and describes how to create a Newton disk in Unity. The practical result will be a disc of 7 colors of the rainbow and will form white when rotated.
Keywords: Disc, Newton, Rainbow

Диск Ньютона - это композиция цветов в форме конусов, расположенных на диске. Когда происходит его вращение очень быстро, цвета сливаются в целый белый диск. Это явление также можно описать как добавление цвета. Когда цвета вращаются так быстро, их отражение будет сливаться друг с другом и создавать высшую сумму - белый цвет, который будет восприниматься человеческими глазами.

Цель статьи воссоздать эффект диска Ньютона в Unity.

С.Д. Лизяев и Р.С. Молотов рассмотрели некоторые особенности в создании анимации для обучающих симуляторов в Unity [1]. Ф.Р.Аметов, И.Ш. Мевлют, М.Р. Абляев подробно рассмотрели важность и необходимость правильного использования анимационных и звуковых элементов при проектировании и разработке игрового компьютерного приложения [2]. В.А. Рогачев, О.А Шабалина расписали сведения о игровом движке Unity и написали рекомендации по его использованию в примере компьютерной игры [3]. М.Р. Абляев, Ф.Р. Аметов, И.Ш. Мевлют описали основные возможности и порядок разработки программного продукта посредством интегрированной среды разработки Unity с использованием технологии виртуальной реальности [4].

Для начала напишем скрипт, который будет реализовать логику работы диска. Так же он будет следить за курсором для вращения диска (рис.1).

```
Vector3 MouseStart;
Vector3 MouseEnd;
public float torqueMultiplier = 12;
public int maxAngularVelocity = 40;
int rotationDirection;

Rigidbody rb;
Material NewtonDiscEffectMat;
```

Рисунок 1 - Логика

Координаты мыши используются для определения силы, вращающей диск. Если бы оно было постоянным, вращение началось бы из смешанного состояния и не было бы возможности просмотреть переход из начального состояния. «TorqueMultiplier» - это константа для увеличения крутящего момента, рассчитываемого по расстоянию между позициями мыши. «maxAngularVelocity» - это переопределение «maxAngularVelocity» игрового объекта. «RotationDirection» используется для представления направления вращения диска. Наконец, «rb» и «NewtonDiscEffectMat» используются для хранения ссылок на компоненты игрового объекта.

По умолчанию «maxAngularVelocity» для «Rigidbody» составляет 7 радиан в секунду. Достаточно достичь правдоподобной скорости, чтобы имитировать поведение диска Ньютона. Итак, первый шаг - увеличить его. Значение «maxAngularVelocity» является общедоступным, поэтому можно настроить его в редакторе (рис.2).

```
void Start()
{
    rb = GetComponent<Rigidbody>();
    rb.maxAngularVelocity = maxAngularVelocity;
    NewtonDiscEffectMat = GetComponent<MeshRenderer>().material;
}
```

Рисунок 2 – Скорость диска

Когда игровой объект вращается, следует применить радиальное размытие. Степень применения смешивания линейна по отношению между текущей угловой скоростью и «maxAngularVelocity» игрового объекта. Для коэффициента, умноженного на направление вращения, в шейдере устанавливается значение переменной «Angle» для повторной визуализации материала.

Когда мышь нажата, начальное положение сохраняется. Когда она отпускается, расстояние между конечной и начальной точками определяет

часть крутящего момента, который необходимо приложить для вращения диска. С какой стороны прикладывать крутящий момент определяется положением «mouse» по оси «x» (рис.3).

```
void Update()
{

    if(rb.angularVelocity.magnitude > 0)
    {
        float ratio = rb.angularVelocity.magnitude / maxAngularVelocity;
        NewtonDiscEffectMat.SetFloat("_Angle", ratio*360*rotationDirection);
    }

    if (Input.GetMouseButtonDown(0))
    {
        MouseStart = Input.mousePosition;
    }

    if (Input.GetMouseButtonUp(0))
    {
        MouseEnd = Input.mousePosition;
        float distance = Vector3.Distance(MouseEnd, MouseStart);
        rotationDirection = (MouseEnd.x>MouseStart.x) ? -1 : 1;
        rb.AddTorque(transform.up * distance * torqueMultiplier * rotationDirection);
    }

}
```

Рисунок 3 –Управление диском

Далее создаем поверхностный шейдер с серфингом, поскольку он является основной функцией. «rotateUV» - это определенная функция для применения поворота на градусы к данной координате «UV» . Это чистое вращение матрицы в системе координат. Координаты «UV» начинаются с 0,0 и заканчиваются 1,1, поэтому часть +-0,5 - это место, где точка перемещается в исходную точку и возвращается в исходное место после применения поворота (рис.4).

В функции серфинга текстура поворачивается с шагом по углу во временах выборки, и значения цвета соответствующей точки суммируются, чтобы обеспечить аддитивную цветовую систему с использованием усредняющего веса (рис.5).

«RotationAngle» - это значение угла, применяемое для каждого шага вращения. Допустим, sample = 5 и angle = 10. Текстура поворачивается на 2 градуса в каждом цикле, всего 5 раз. «sampleWeight» - это средний вес для каждого расчета вращения. Результирующее вращение сохраняется в «FragColor», что в совокупности приводит к добавлению цвета (рис.6).

```
Shader "Custom/SpinBlur"{
        Properties{
                _Color("Main Color", Color) = (1,1,1,1)
                _Samples("Samples", Range(0,360)) = 100
                _Angle("Angle", Range(-360,360)) = 10
                _MainTex("Color (RGB) Alpha (A)", 2D) = "white"
        }
                SubShader{
                Tags{ "Queue" = "Transparent" "RenderType" = "Transparent" }

                LOD 200
                Cull Off

                CGPROGRAM
                #pragma target 3.0
                #pragma surface surf Lambert alpha

                sampler2D _MainTex;
                int _Angle;
                int _Samples;
                float4 _Color;

                struct Input {
                        float2 uv_MainTex;
                        float4 screenPos;
                };

                float2 rotateUV(float2 uv, float degrees) {
                        const float Deg2Rad = (UNITY_PI * 2.0) / 360.0;
                        float rotationRadians = degrees * Deg2Rad;
                        float s = sin(rotationRadians);
                        float c = cos(rotationRadians);
                        float2x2 rotationMatrix = float2x2(c, -s, s, c);
                        uv -= 0.5;
                        uv = mul(rotationMatrix, uv);
                        uv += 0.5;
                        return uv;
                }
```

Рисунок 4 - Шейдер

```
42
43                  void surf(Input IN, inout SurfaceOutput o) {
44                          float2 vUv = IN.uv_MainTex;
45                          float2 coord = vUv;
46                          float4 FragColor = float4(0.0, 0.0, 0.0, 0.0);
47                          int samp = _Samples;
48                          if (samp <= 0) samp = 1;
49                          float rotationAngle = (float)_Angle / (float)samp;
50                          float sampleWeight = 1.0 / samp;
51                          for (float i = 0; i < samp; i++) {
52                                  coord = rotateUV(coord, rotationAngle);
53                                  float4 texel = tex2D(_MainTex, coord);
54                                  texel *= sampleWeight;
55                                  FragColor += texel;
56                          }
```

Рисунок 5 - Шейдер

```
57                          float4 c = FragColor * _Color;
58                          o.Albedo = c.rgb;
59                          o.Alpha = c.a;
60                  }
61                  ENDCG
62          }
63                  FallBack "Diffuse"
64  }
```

Рисунок 6 - Шейдер

В конце получается вот такой диск (рис.7).

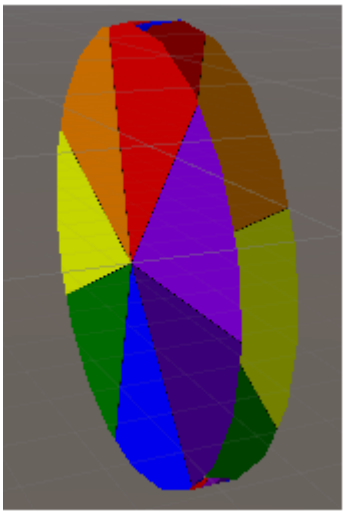

Рисунок 7 – макет диска

Теперь осталось проверить работоспособность диска, после его запуска получается немного розоватый цвет, но все же из-за графики цвет не получается белым (рис.8).

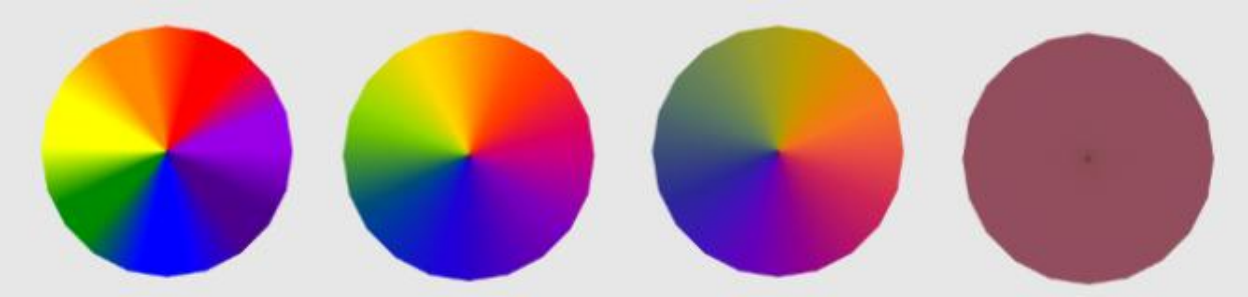

Рисунок 8 – работа диска

В данной статье была рассмотрена возможность реализации диска Ньютона в Unity. Практическим результатом является скрипт рабочего диска Ньютона.

Библиографический список

1. Лизяев С.Д., Молотов Р.С. Особенности создания анимации при разработке обучающих симуляторов в среде Unity 3D// Современные научные исследования и инновации. 2017. № 7-5 (43). С. 47-55.
2. Аметов Ф.Р., Мевлют И.Ш., Абляев М.Р. Инструменты для создания анимационного и звукового сопровождения компьютерной игры на платформе unity 3d// Труды Международного симпозиума «Надежность и качество». 2017. №6. С. 14-20.
3. Рогачев В.А., Шабалина О.А. Разработка компьютерных игр в среде Unity 3D: основные приемы работы и примеры их применения // Автоматика. Вычислительная техника. 2019. №1. С. 41-43
4. Абляев М.Р., Аметов Ф.Р., Мевлют И.Ш. Unity 3D как средство разработки программ с возможностью визуализации технологии виртуальной реальности// Труды Международного симпозиума «Надежность и качество». 2018. №9. С. 43-52.