Title: Collision of multiple objects

EXPERIMENT NO: 10 **Name:** SHAKYA JAY JITENDRA

Date: 8-9-2025 **Reg. No:** 24BCG10123

Aim: To study the effect of collisions between multiple objects in Unity physics, and observe how they exchange velocity and change direction in Unity.

Description/Concept:

The entire simulation is governed by two immutable laws of physics:

Conservation of Momentum: The C# script's core calculation ensures that for any sphere-to-sphere collision, the total vector momentum (p=mv) of the two-sphere system remains constant. Momentum is transferred between the spheres, but the system's total "quantity of motion" is preserved.

Conservation of Kinetic Energy: By defining the collisions as perfectly elastic (via the Physic Material with Bounciness = 1), the simulation also upholds the conservation of kinetic energy (K=21 mv2). No energy is lost to heat, sound, or object deformation, allowing the motion to continue indefinitely.

The complex vector formula inside the ElasticCollision.cs script is the direct mathematical solution derived from applying these two conservation laws simultaneously.

Implementation and Outcome

The project successfully implements these principles by creating a closed system—the box—where energy and momen cannot escape. It intelligently separates two types of interactions:

Sphere-to-Sphere: Handled by our custom C# script, which accurately computes the outcome of a complex, multi-body dynamic interaction.

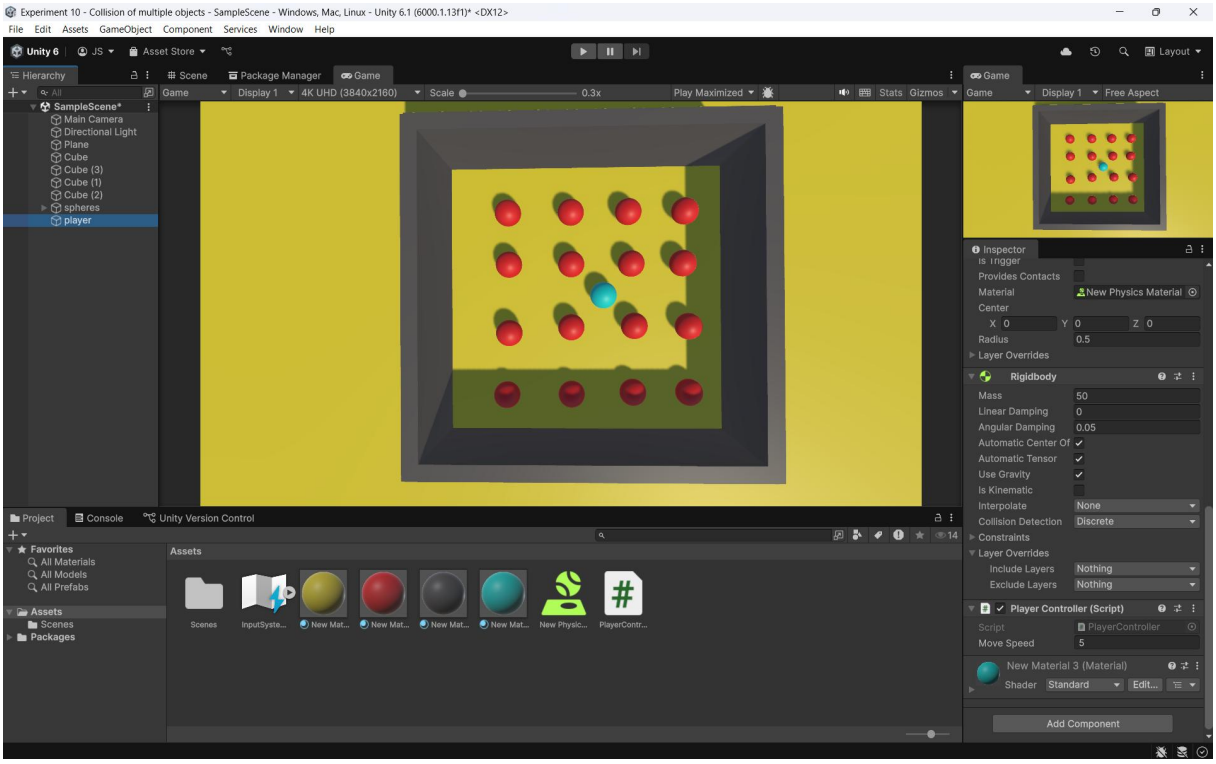
Sphere-to-Wall: Handled by Unity's robust physics engine, guided by the simple rules defined in the Physic Material.

Program/Coding:

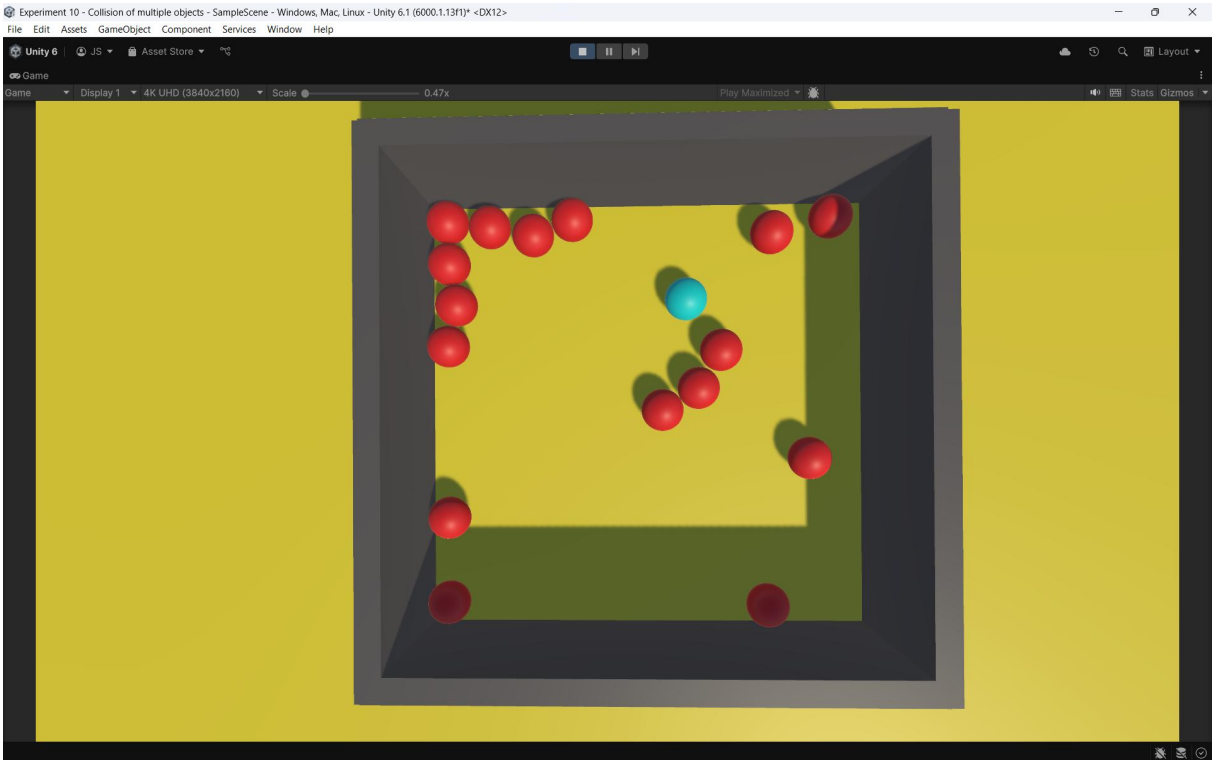
```
using UnityEngine;


public class PlayerController : MonoBehaviour
{
    public float moveSpeed = 5f;  // Movement speed
    private Rigidbody rb;
```

```
    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        // Get input
        float moveX = Input.GetAxis("Horizontal");  // A/D or Left/Right
        float moveZ = Input.GetAxis("Vertical");    // W/S or Up/Down

        // Movement vector
        Vector3 movement = new Vector3(moveX, 0f, moveZ) * moveSpeed;

        // Apply movement using physics
        rb.velocity = new Vector3(movement.x, rb.velocity.y, movement.z);
    }
}}
```

Output:



```
```



Conclusion:

In conclusion, this project is more than just a visual animation; it is a functioning physics simulation. It effectively translates theoretical physics into computational logic, demonstrating how fundamental laws can be used to model complex, emergent behavior. This serves as a foundational example of how realistic interactions in video games, engineering simulators, and scientific

Result:

Script was created for Collision of multiple objects were successfully implemented and tested in Unity Game Engine 6.1 Their application in game mechanics was understood through coding and console outputs.