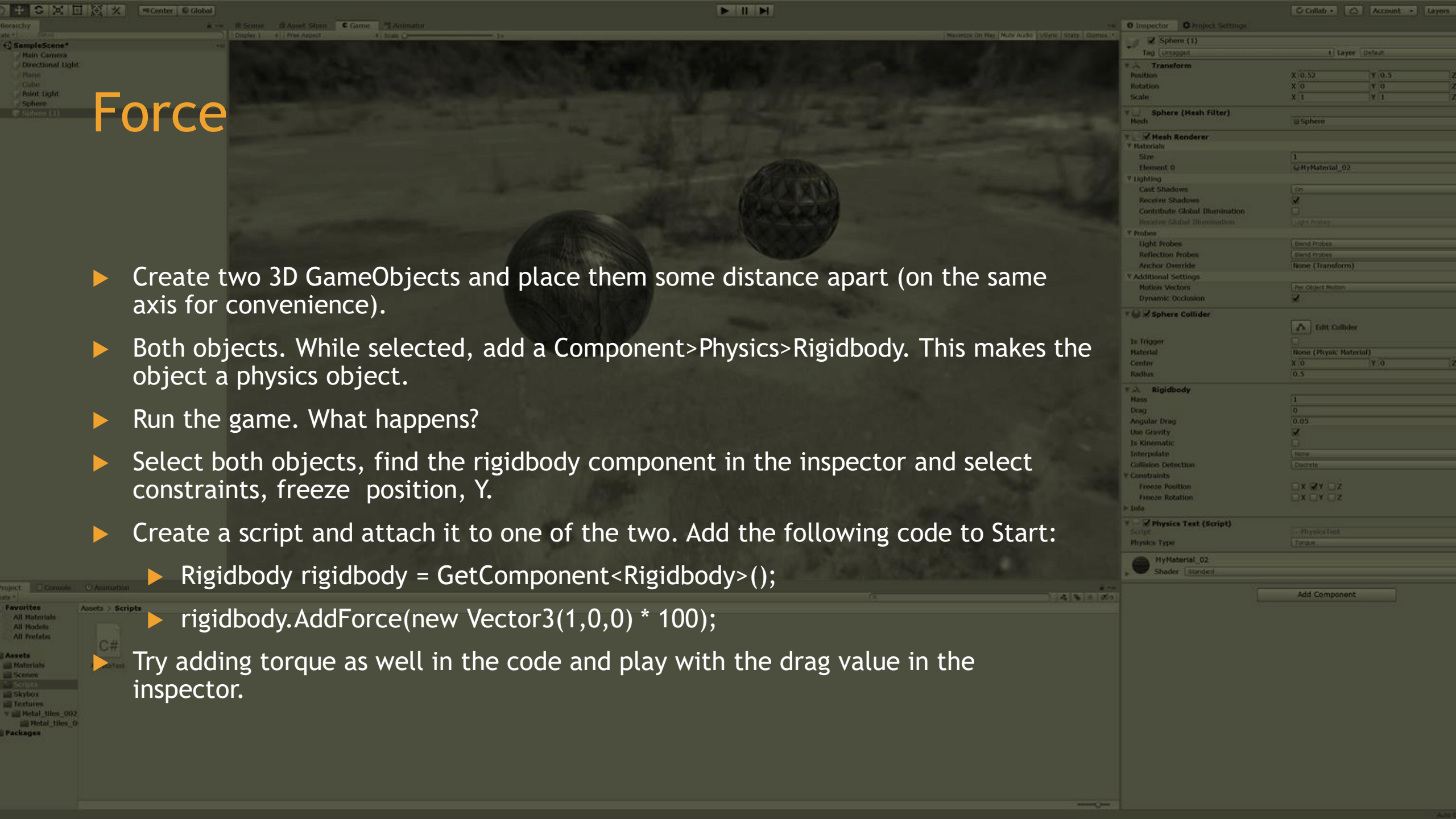
Unity 3D

Physics

Force



- Create two 3D GameObjects and place them some distance apart (on the same axis for convenience).
- Both objects. While selected, add a Component>Physics>Rigidbody. This makes the object a physics object.
- Run the game. What happens?
- Select both objects, find the rigidbody component in the inspector and select constraints, freeze position, Y.
- Create a script and attach it to one of the two. Add the following code to Start:
  - Rigidbody rigidbody = GetComponent<Rigidbody>();
  - rigidbody.AddForce(new Vector3(1,0,0) * 100);
- Try adding torque as well in the code and play with the drag value in the inspector.

```
rializeField] private PhysicsType physicsType;
rializeField] private bool useOnCollision;

ivate Rigidbody rigidbody;

eferences
ivate enum PhysicsType

    Force,
    Torque
}

// Start is called before the first frame update
0 references
void Start()
{
    rigidbody = GetComponent<Rigidbody>();

    switch (physicsType) {
        case PhysicsType.Force:
            {
                rigidbody.AddForce(new Vector3(1, 0, 0) * 100);
                break;
            }
        case PhysicsType.Torque:
            {
                rigidbody.AddTorque(new Vector3(1, 0, 0) * 100);
                break;
            }
    }
}

1 reference
public void Jump()
{
    rigidbody.constraints = RigidbodyConstraints.None;
    rigidbody.AddForce(new Vector3(0, 1, 0) * 100);
}

0 references
void OnCollisionEnter(Collision collision)
{
    if (!useOnCollision)
    {
        return;
    }

    Rigidbody collisioRigidbody = collision.gameObject.GetComponent<Rigidb
    if (collisioRigidbody)
    {
        collisioRigidbody.velocity = Vector3.zero;
        collisioRigidbody.angularVelocity = Vector3.zero;
    }
    collision.gameObject.GetComponent<PhysicsTest>()?.Jump();
    Destroy(collision.gameObject, 0.5f);
}
```

Collisions

- Add this function to your script and run:
  - void OnCollisionEnter(Collision collision)

    {

    Destroy(collision.gameObject, 0.5f);

    }
  - What does 0.5f do?
- Add a script to the other game object. Add this code to it:
  - public void Jump() {

    private Rigidbody rigidbody = GetComponent<Rigidbody>();

    rigidbody.constraints = RigidbodyConstraints.None;

    rigidbody.AddForce(new Vector3(0, 1, 0) * 100); }
- Then change the other script's collision code to:
  - void OnCollisionEnter(Collision collision) {

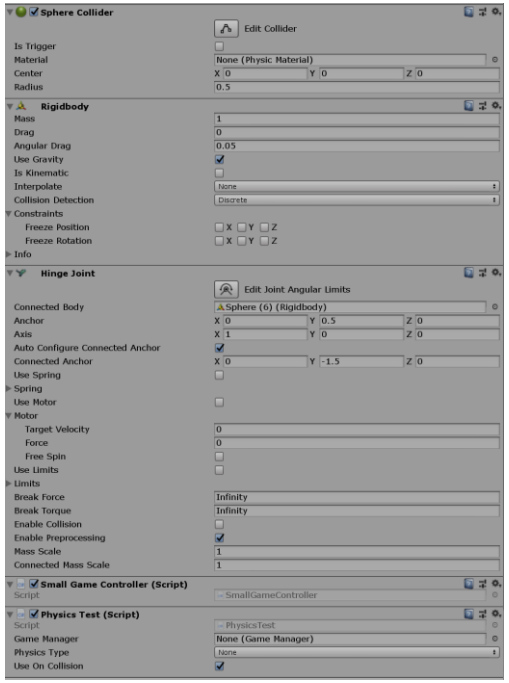
    collision.gameObject.GetComponent<…>().Jump();

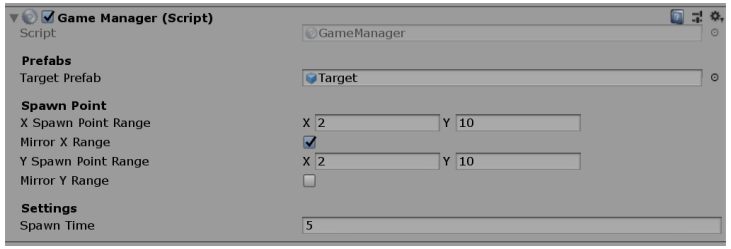
    Destroy(collision.gameObject, 0.5f);

  }
- This pattern is very powerful. A surprising number of games can be built with just this!

Hinges

- Add another game object with a rigidbody attached.
- While the object is selected: Component > Physics > Hinge Joint
- Add one of the other existing game objects as the "Connected Body" and run.





A Small Game

- Create a game object and make its position fixed using constraints.
- Add 5-10 other rigidbody objects with hinges and make a chain attached to the fixed one.
- Attach a script to the last object on the chain.
- When a button is pressed, apply a force to the last object. Pick a force direction that does something interesting.
- Create a target object (e.g. a sphere) and add a script to it. When this object is hit it should be destroyed. Save it as a prefab.
- Instantiate random targets using a script. See this on how to instantiate:
  - https://docs.unity3d.com/Manual/InstantiatingPrefabs.html
  - The script that instantiates the targets can be placed on an empty game object.
- Keep track of how many targets you hit with the chain.

StringRepo.cs | GameManager.cs | SmallGameController.cs | PhysicsTest.cs

Assembly-CSharp

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

3 references
public class StringRepo
{
    public static string TagetTag = "Target";
    public static string ImmortalTag = "Immortal";
}
```

StringRepo.cs | GameManager.cs | SmallGameController.cs | PhysicsTest.cs

Assembly-CSharp

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

0 references
public class SmallGameController : MonoBehaviour
{
    private Rigidbody rigidbody;
    // Start is called before the first frame update
    0 references
    void Start()
    {
        rigidbody = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        if (Input.GetButtonDown("Fire1"))
        {
            //Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            //if (Physics.Raycast(ray))
            //    Instantiate(particle, transform.position, transform.rotation);

            Vector2 screenMidPoint = new Vector2(Screen.width / 2f, Screen.height / 2f);
            Vector2 mousePosition = Input.mousePosition;

            Vector2 direction = mousePosition - screenMidPoint;
            direction.Normalize();
            Vector3 direction3D = new Vector3(direction.x, direction.y, 0f);

            rigidbody.AddForce(direction3D * 1000);
        }
    }
}
```

StringRepo.cs | GameManager.cs | SmallGameController.cs | PhysicsTest.cs

Assembly-CSharp

```cs
1 reference
public class GameManager : MonoBehaviour
{
    [Header("Prefabs")]
    [SerializeField] private GameObject targetPrefab;

    [Header("Spawn Point")]
    [SerializeField] private Vector2 xSpawnPointRange;
    [SerializeField] private bool mirrorXRange;
    [SerializeField] private Vector2 ySpawnPointRange;
    [SerializeField] private bool mirrorYRange;

    [Header("Settings")]
    [SerializeField] private float spawnTime;

    private float remainingSpawnTime = 0;
    2 references
    public float CollectedTargets { get; private set; }

    // Update is called once per frame
    0 references
    void Update()
    {
        // Spawn target and reset timer
        if(remainingSpawnTime <= 0)
        {
            SpawnTarget(PickRandomPoint());
            remainingSpawnTime = spawnTime;
        }
        else
        {
            remainingSpawnTime -= Time.deltaTime;
        }

        if (Input.GetKeyDown(KeyCode.S))
        {
            print(CollectedTargets);
        }
    }

    /// <summary>
    /// Spawn target in a point
    /// </summary>
    1 reference
    private void SpawnTarget(Vector2 spawnPoint)
    {
        GameObject go = Instantiate(targetPrefab, new Vector3(spawnPoint.x, spawnPoint.y, 0f), Quaternion.identity);
        go.GetComponent<PhysicsTest>().gameManager = this;
        go.tag = StringRepo.TagetTag;
    }

    /// <summary>
    /// Pick a random 2D point based on options
    /// </summary>
    /// <returns></returns>
    1 reference
    private Vector2 PickRandomPoint()
    {
        float x = Random.Range(xSpawnPointRange.x, xSpawnPointRange.y);
        float y = Random.Range(ySpawnPointRange.x, ySpawnPointRange.y);

        if (mirrorXRange)
        {
            int mirror = Random.Range(0, 2);
            if (mirror == 1)
            {
                x *= -1;
            }
        }

        if (mirrorYRange)
        {
            int mirror = Random.Range(0, 2);
            if (mirror == 1)
            {
                y *= -1;
            }
        }

        return new Vector2(x, y);
    }

    /// <summary>
    /// Add "value" collecred targets
    /// </summary>
    /// <param name="value"></param>
    1 reference
    public void AddCollectedTargets(int value = 1)
    {
        CollectedTargets += value;
    }
}
```

StringRepo.cs | GameManager.cs | SmallGameController.cs | PhysicsTest.cs

Assembly-CSharp

```cs
public class PhysicsTest : MonoBehaviour
{
    [Tooltip("Use it only on small game")]
    public GameManager gameManager;

    [SerializeField] private PhysicsType physicsType;
    [SerializeField] private bool useOnCollision;

    private Rigidbody rigidbody;

    3 references
    private enum PhysicsType
    {
        Force,
        Torque,
        None
    }

    // Start is called before the first frame update
    0 references
    void Start()
    {
        rigidbody = GetComponent<Rigidbody>();

        switch (physicsType) {
            case PhysicsType.Force:
                {
                    rigidbody.AddForce(new Vector3(1, 0, 0) * 100);
                    break;
                }
            case PhysicsType.Torque:
                {
                    rigidbody.AddTorque(new Vector3(1, 0, 0) * 100);
                    break;
                }
        }
    }

    1 reference
    public void Jump()
    {
        rigidbody.constraints = RigidbodyConstraints.None;
        rigidbody.AddForce(new Vector3(0, 1, 0) * 100);
    }

    0 references
    void OnCollisionEnter(Collision collision)
    {
        if (!useOnCollision)
        {
            return;
        }

        Rigidbody collisioRigidbody = collision.gameObject.GetComponent<Rigidbody>();
        if (collisioRigidbody)
        {
            collisioRigidbody.velocity = Vector3.zero;
            collisioRigidbody.angularVelocity = Vector3.zero;
        }
        collision.gameObject.GetComponent<PhysicsTest>()?.Jump();
        if (collision.gameObject.tag != StringRepo.ImmortalTag)
        {
            Destroy(collision.gameObject, 0.5f);
        }
    }

    /// <summary>
    /// If the gameobject have tag as target then +1 on collected targets in gamemanager
    /// </summary>
    0 references
    private void OnDestroy()
    {
        if(tag == StringRepo.TagetTag)
        {
            if (gameManager != null)
            {
                gameManager.AddCollectedTargets();
            }
        }
    }
}
```

75 %