Tutorial – Unity Networking 4: Server-side Objects

Complete the Networking Unity Animation tutorial before doing this one. It can happen after or before Networking Events.

So far we have a fairly sophisticated entity that can animate, fire laser beams and lose health over the network. But both entities in the game are being controlled by a client and having their activity broadcast by the host. Here we'll look at independent objects that update on the server, and broadcast to all clients.

This is the key to making networked AI entities. For now we'll start with a simple projectile. These will need to spawn, destroy themselves, and keep updating their position and let all clients know about these events.

Networked Projectiles

We can replace the laserbeam LineRenderer and raycast with a separate projectile GameObject that travels in a straight line and uses a Trigger to collide with enemies and damage them.

In a single player game we'd spawn the projectile and give it a limited lifespan, destroying it either at the end of the lifespan or when it hits something.

In a networked situation we want to make sure that the spawn and destroy are synched across the network.

Create a Bullet prefab as a scaled down Sphere that has a NetworkTransform (and thus a NetworkIdentity). Set the collider to be a Trigger, add a kinematic Rigidbody, and give it a new Bullet script which we'll derive from Mirror.NetworkBehaviour

Give it a Vector3 velocity member and an Update where it uses this velocity to travel.

```
void Update()
    {
        transform.position += velocity * Time.deltaTime;
    }
```

Give it a OnTriggerEnter function where it subtracts health from any suitable object it hits, and also destroys itself on any collision.

```
private void OnTriggerEnter(Collider other)
    {
        Health health = other.GetComponent<Health>();

        // this will get transmitted via a SyncVar
        if (health)
            health.health -= damage;

        // destroy us when we hit something
        Destroy(gameObject);
    }
```

In Start() we can schedule the objects destruction after a specified lifetime if it never hits anything

```
    public void Start()
    {
        Destory(gameObject, lifetime);
    }
```

Finally, modify the LaserBeam script to spawn a projectile instead of turning on the LineRenderer and doing a raycast. We can do this in the Hit()n function which is called in response to an AnimationEvent, instead of calling DoLaser() like we were before.

```
    public Bullet bulletPrefab;

….

    public void Hit()
    {
        // this gets called in response to animation events
        // DoLaser();

        GameObject go = Instantiate(bulletPrefab.gameObject,
            transform.position + Vector3.up + transform.forward,
            Quaternion.LookRotation(transform.forward));
        Bullet bullet = go.GetComponent<Bullet>();
        bullet.velocity = transform.forward * 5;
    }
```

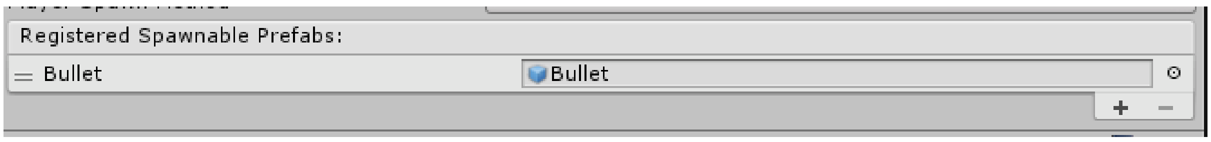
Put the Bullet prefab into the LaserBeam's bulletprefab member in Unity, and you should have a working single player projectile now. Pressing CTRL will fire off a small projectile that travels across the map for a certain distance and destroys itself if it hits an object.

Testing this on a single host should work.
We now need to network enable it. As in the C++ tutorials, the Server should have the responsibility of spawning the projectile and notifying all clients that there's a new object in the scene. The player (who has authority to do so) can notify the server that they've just fired.

Similarly, the server should be responsible for detecting a collision, subtracting health if required, and telling all clients that the bullet has despawned now.

In order to spawn an object across the network, we need to assign it to the NetwrokManager's list of spawnable objects. Remember that every network message must consist of Plain Old Data, and thus we can't send a prefab reference across the network. We can send an index into an array which exists in the scene data though.

Add the Bullet prefab to the NetworkManager's list of spawnable obejcts.


Registered Spawnable Prefabs:

Bullet	Bullet

We can now make the spawning code only happen on the Server, and let the Server tell all clients that the spawn has happened.

We do this by setting the Hit() function to only occur on the server using Mirror's Server attribute. After instantiating the prefab and setting its values, we call NetworkServer.Spawn to transmit the object's existence and state to all clients.

```
    [Server]
    public void Hit()
    {
        // this gets called in response to animation events
        // DoLaser();

        GameObject go = Instantiate(bulletPrefab.gameObject,
            transform.position + Vector3.up + transform.forward,
            Quaternion.LookRotation(transform.forward));
        Bullet bullet = go.GetComponent<Bullet>();
        bullet.velocity = transform.forward * 5;

        NetworkServer.Spawn(bullet.gameObject);
    }
```

In the bullet, we want to update the OnTriggerEnter function to only happen on the server, and to call a networked despawn function instead of the simple GameObject.Destroy we called earlier.

```
    [Server]
    private void OnTriggerEnter(Collider other)
    {
        Health health = other.GetComponent<Health>();

        // this will get transmitted via a SyncVar
        if (health)
            health.health -= damage;

        // destroy us when we hit something
        NetworkServer.Destroy(gameObject);
    }
```

Finally, we need to modify the Start function so that when we spawn the object on the server it queues up a networked destroy after its lifetime has expired. Replace the existing Start() function with an override to one of Mirror's functions that only gets called on server. NetworkServer.Destroy doesn't allow for a delayed version, so we have to do this mnaully with invoke() or could use a Coroutine.

```
    public override void OnStartServer()
    {
        Invoke(nameof(DestroySelf), lifeTime);
    }

    // destroy for everyone on the server
    [Server]
    void DestroySelf()
    {
        NetworkServer.Destroy(gameObject);
    }
```

Add the [Server] tag to the Update function and we're done. We no have bullets that spawn, move and despawn on the server. The server notifies clients of any spawn or destruction events via the NetwrokServer static functions we've used.

The server moves the bullets and they are updated by the NetworkTransform component.

(This is a bit of overkill, but, it paves the way for a pattern where we could have homing missiles that move intelligently, up to fullblown decision making AI. Look at the tanks example in Mirror's code for an alternative where the projectile is a RigidBody and we rely on the physics engine to move bullets locally and independently on each client)

When the server detects a collision with a bullet and a target it despawns the bullet via the NetworkServer, and modifies the target's health which updates on all clients as a SyncVar with a hook function to cause blood splatter and health bar update.