```
    private void ShootFinger(Finger finger, Vector3 targetPoint)
    {
        // cast a ray from the fingerSlot to the 'targetPoint' (or just get direction)
        Vector3 directionToShoot = (targetPoint - finger.Slot.position).normalized;

        // shoot the actual rigidbody of the finger
        GameManager.Instance.PhysicsController.ColliderBool(finger.ColliderFinger, true);
        GameManager.Instance.PhysicsController.RigidbodyBoolKinematic(finger.RigidbodyFinger, false);
        GameManager.Instance.PhysicsController.RigidbodySetVelocity(finger.RigidbodyFinger, directionToShoot *
GameManager.Instance.PhysicsController.ShootingVelocity);

        // update lists
        ShotFingers.Add(finger);
        StationedFingers.Remove(finger);
        GameManager.Instance.FingerPathManager.AddToShotList(finger);

        // play muzzle particle & activate trail
        GameManager.Instance.ParticleManager.PlaySimpleParticle(finger.ParticleMuzzle);
        GameManager.Instance.ParticleManager.SetTrailStatus(finger.TrailsShot, true);

        // play audio shot
        GameManager.Instance.AudioController.PlayRandomAudio(_soundsFingerShot, false, 0);
    }
    public void OnFingerHitSomething(Finger fingerThatHitSomething, Collision collision)
    {
        GameObject hitObject = collision.gameObject;
        Vector3 impactVelocity = fingerThatHitSomething.RigidbodyFinger.velocity;

        // freeze the finger in place (maybe even parent it, only if it's shakeable ?)
        GameManager.Instance.PhysicsController.RigidbodySetVelocityZero(fingerThatHitSomething.RigidbodyFinger);
        GameManager.Instance.PhysicsController.RigidbodyBoolKinematic(fingerThatHitSomething.RigidbodyFinger, true);
        GameManager.Instance.PhysicsController.ColliderBool(fingerThatHitSomething.ColliderFinger, false);

        // impact and charging logic
        GameManager.Instance.MagnetismController.FingerHitTryCharge(fingerThatHitSomething, collision, hitObject, impactVelocity);
        // particle hit
        GameManager.Instance.ParticleManager.PlaySimpleParticle(fingerThatHitSomething.ParticleImpact);
        // sound
        GameManager.Instance.AudioController.PlayAudio(_soundFingerHit, false, 0, fingerThatHitSomething.transform);
    }
    public void ClapHands()
    {
        if (GameManager.Instance.IsPaused == true)
        {
            return;
        }
        // possibly change the logic for when I can clap and magnetize...
        if (_currentFingerCountLeft < 5 || _currentFingerCountRight < 5)
        {
            _canClap = true;
        }
        else
        {
            _canClap = false;
        }

        if (_canClap == true && _isClapping == false)
        {
            _isClapping = true;

            // reset the look rotation of the hands and animate them
            StartCoroutine(ResetHandRotationRoutine(_handLeft));
            StartCoroutine(ResetHandRotationRoutine(_handRight));
            _handLeft.Palm.AnimatePalmCall(_handLeft.Palm.AnimClap);
            _handRight.Palm.AnimatePalmCall(_handRight.Palm.AnimClap);

            // stretch any fingers that are still present on the hand
            for (int i = 0; i < StationedFingers.Count; i++)
            {
                StationedFingers[i].AnimateFingerCall(StationedFingers[i].AnimStretched);
            }

            // activate actual magnetism
            StartCoroutine(GameManager.Instance.MagnetismController.MagnetismRoutine());
            StartCoroutine(ReturnFingersAndResetHandsRoutine());

            // play audio clap
            GameManager.Instance.AudioController.PlayRandomAudio(_soundsClap, false, (TimeToReachClap - 0.09f));
        }
    }
```

```
80
81
82
83      public IEnumerator RetractFingerRoutine(Finger fingerToRetract)
84      {
85          // unparent + reset scale
86          GameManager.Instance.TransformController.SetFingerTransformValuesForReturn(fingerToRetract);
87
88          // stop any velocity/physics related to this finger
89          GameManager.Instance.PhysicsController.RigidbodySetVelocityZero(fingerToRetract.RigidbodyFinger);
90          GameManager.Instance.PhysicsController.RigidbodyBoolKinematic(fingerToRetract.RigidbodyFinger, true);
91          GameManager.Instance.PhysicsController.ColliderBool(fingerToRetract.ColliderFinger, false);
92
93          // disable shot trail & enable return trail
94          GameManager.Instance.ParticleManager.SetTrailStatus(fingerToRetract.TrailsShot, false);
95          GameManager.Instance.ParticleManager.SetTrailStatus(fingerToRetract.TrailsReturn, true);
96
97          // create initial path & keep updating the path on this finger (in manager)
98          GameManager.Instance.FingerPathManager.CreateFirstFramePath(fingerToRetract);
99          GameManager.Instance.FingerPathManager.EnablePathUpdating(true);
100
101         // set the speed on the pathFollower
102         GameManager.Instance.FingerPathManager.SetSpeedFingerPath(fingerToRetract);
103
104         // play sound reconnection (with delay)
105         GameManager.Instance.AudioController.PlayAudio(_soundFingerReconnect, false, _fingerReturnTime - 0.1f);
106
107         // once the finger has arrived...
108         yield return new WaitForSeconds(FingerReturnTime);
109
110         GameManager.Instance.FingerPathManager.ResetAndDisablePath(fingerToRetract);
111
112         ReconnectFingerVisually(fingerToRetract);
113
114         // update lists and values
115         GameManager.Instance.FingerPathManager.RemoveRetractFingerPath(fingerToRetract);
116
117         ReplenishFinger(fingerToRetract);
118     }
119
120
121     private void ResetHandsAndFingersToIdle()
122     {
123         _handRight.Palm.AnimatePalmTrigger(_handRight.Palm.TriggerIdle);
124         _handLeft.Palm.AnimatePalmTrigger(_handLeft.Palm.TriggerIdle);
125         for (int i = 0; i < StationedFingers.Count; i++)
126         {
127             var fingerOfInterest = StationedFingers[i];
128             fingerOfInterest.AnimateFingerCall(fingerOfInterest.AnimIdle);
129         }
130     }
131
132     private void ReconnectFingerVisually(Finger fingerThatReturned)
133     {
134         fingerThatReturned.Reconnect();
135         fingerThatReturned.AnimateFingerCall(fingerThatReturned.AnimRecover);
136         GameManager.Instance.ParticleManager.SetTrailStatus(fingerThatReturned.TrailsReturn, false);
137     }
138
139     private void ReplenishFinger(Finger fingerToReplenish)
140     {
141         ShotFingers.Remove(fingerToReplenish);
142         StationedFingers.Add(fingerToReplenish);
143
144         if (fingerToReplenish.Hand.TypeHand == HandType.Left)
145         {
146             _currentFingerCountLeft += 1;
147         }
148         else
149         {
150             _currentFingerCountRight += 1;
151         }
152     }
153
154
155
```