The **WonderBuffaloTests Unity project** has 2 scenes

- test_scene
  This is just a sandbox for my tests.
  You will find 2 objects shape_before and shape_after. This setup explains how to morph an object into another.
  The shape_before object has a **ShapeWarperScript** component and a **BeforeMaterial** with a ShapeWarperShader
  ShapeWarperScript turns the object into a sphere with radius decreasing with time till it reaches 0 and then morphs the other object from a sphere of radius 0 to the final shape.
  The **ShapeWarperScript** has 3 parameters
  - *Other [GameObject]*
    - It is the second object the geometry should morph into.
  - *Radius [UnityUnit]*
    - It is the maximum radius of the sphere
  - *Duration [sec]*
    - it is the time from the initial shape to the final shape

- test_leapmotion
  This scene is fully setup for leap motion.
  You ll need a hmd and a leap motion camera http://store-us.leapmotion.com/
  I packaged the leap motion assets necessary to replicate the setup I used. You'll find the package **WonderBuffalo_LeapMotion.unitypackage** in the repository root. Import all of that into your project and you should be good to go in terms of Leap Motion scripts and assets.
  I used in particular the **Leap Motion Interaction Engine** module for the collision/interaction with objects. I noticed that by importing this package your project will stop building in visual studio, but you can still attach it to unity and script debugging works just fine so I guess we can live with that. :)

  This scene allows you to interact with some object on the floor:
  - a capsule and a sphere which you can move and grab with your leap motion hands
    - Info on how to setup leapmotion in your unity project are here.
      - https://developer.leapmotion.com/documentation/unity/devguide/Project_Setup.html
    - Info on how to set up Interaction Engine scene are here.
      - https://developer.leapmotion.com/documentation/unity/unity/Unity_IE_Setup.html
      - https://developer.leapmotion.com/documentation/unity/unity/Unity_IE_Layers.html
      - https://docs.unity3d.com/Manual/Layers.html
    - After the setup to do something similar in your project

        - Add a box or sphere collider to your game object. **[Very Important] Other collision primitives are not currently properly supported by the Interaction Engine as per my tests!**
        - Add a Leap Motion Interaction Behaviour script and set the interaction manager
        - Set the object to be in the **Interaction layer** as recommended in the docs otherwise it won't interact with leap motion hands
    - Please notice how the 2 objects have the **GlowSurfaceShader** that I wrote attached to them
        - Parameters
            - *Color*
                - tint applied to the main texture
            - *Albedo*
                - main texture
            - *Rim Color*
                - glow color
            - *Rim Power*
                - smaller values make the game object less glowy along the rim. Play with it!
- an Interactive Wall where you can create texture switching ripples.
  Please notice how it has
    - a rigid body
    - a box collider
    - an **InteractivePlaneScript** which extends the **GridScript** - which I wrote.
    - **GridScript** generates waves with arbitrary amplitudes, frequencies and dumpenings.
    - Some notes about the terminology used in the code for the wave
        - *Amplitude [UnityUnit]*
            - this value is multiplied by a factor proportional to the weight of the animal at startup and represents the amplitude of the waves
        - *Spatial decay [UnityUnit]*
            - this value (it must be always negative) is used to exponentially dampen the amplitude of the wave based on the distance from the center. Smaller values will increase the spatial extent of the wave
        - *Temporal decay [sec]*
            - this value(it must be always negative) is used to exponentially dampen the amplitude of the wave based on the time from the beginning of the wave. Smaller values will increase the temporal duration of the wave
        - *Temporal omega [sec^{-1}]*

        - it's 2PI divided by the wave time period. Larger values mean higher temporal frequency
    - *Spatial omega [UnityUnit^{-1}]*
        - it's 2PI divided by wave spatial period. Larger values mean higher spatial frequency
    - *Offset*
        - An extra parameter in the argument of the sinusoid. Usually set to 0

- **Important: the deformation is always applied along the y in object reference framework.** The **InteractivePlaneScript** just add the collision callback to **GridScript** for the interactive wall. When there is a collision with the leap motion hands it spawns a ripple.
    - Parameters
        - *Max Clamp[UnityUnit]*
            - this is used to guide the transition between the texture. Essentially I take the offset along the object y axis due to the waves clamp it between -MaxClamp and +MaxClamp, normalize it between -1 and 1 by dividing by MaxClamp get the absolute value and use that to blend between the textures. This parameter allows you to affect the texture transition easily, making it subtle or dramatic.
        - *Max Ripples*
            - maximum number of ripples supported by the game object.
        - *Time threshold[sec]*
            - this threshold is currently unused but it will basically allow you to decide after what amount of time a wave should be destroyed or chosen to be destroyed in case a new ripple is requested. This could be used to improve performance.
        - *Resolution[pixel]*
            - this is the size of the square texture generated to apply the deformation. At the moment only square textures are supported. 256 is a good value.
- an Interactive Behaviour Script (from Leap Motion) to allow interaction with the leap motion hands
- a shader set to **Unlit/GridShader** - which I wrote
- a **Grid Material** which is used to apply the grid shader to the game object
    - Parameters
        - *Total Amplitude[UnityUnit]*
        - *Max Clamp[UnityUnit]*

- *Offset Direction*
- *World Position offsets*
  - you shouldn't care of these 4 because they are used by the grid script to drive the wave generation
- *Texture Before*
  - this is the texture that is used before the wave switch takes place. The realistic texture in your case.
- *Texture After*
  - this is the texture that is used after the wave switch takes place. The cartoony texture in your case.

This is the setup that must be repeated for each wall in your project where you want the ripple effect.

- a cube which starts behaving as a balloon once your Leap Motion hands collide with it.
  Please notice how it has
  - a rigid body
  - a box collider
  - a **FloatingObjectScript** - which I wrote. This can be used to implement the letter shaped balloon effect. I basically start applying a constant force from the moment of the first collision with the leap motion hands. Collisions are still possible on the game object after it starts floating. I also disable gravity and freeze rotation on the object at startup.
    - Parameters
      - *Acceleration[UnityUnit / sec^2]*: the acceleration that the object experience along the force direction. 0.1 seems a good value.
      - *Force Direction*: the direction of the constant force applied to the object. It should be (0,1,0,1)
  - an Interactive Behaviour Script (from Leap Motion) to allow interaction with the leap motion hands

- TODO list
  - Filter collisions with Interactive Wall to be able to generate ripples in a more user friendly way.