HexLev Control Software

Generated by Doxygen 1.10.0

Chapter 1

HexLev Control Software

The HexLev Control Center is a protoype software developed in Unity game engine to aid the control of the levitator.

The simplified UI acts an interface between the user and the levitator, abstracting Serial and SPI comms involving the Arduino and FPGAs.

The software allows for the creation, deletion and movement of mutliple particles along user-defined trajectories in 3D.

1.0.0.1 Operation

The current version of the software is a prototype and uses a crude implementation for a movement algortihm (see documentation for more details)

The user is at liberty to build the program and run it as a standalone application, however it may be useful to use the program within the Unity editor as this provides users with further technical insight. Pressing the 'Play' button within the Unity editor runs the program.

The UI allows for common user input such as mouse clicks, dragging and scrolling. Devices lIke joysticks and controllers may be used as Unity provides basic functionality, though it is not recommended in this case.

1.0.0.2 Notes

The current prototype version snaps particles to positons, employing a rigid 0-100% scheme. The code will be updated with a solver for the final product.

The Arduino serial code contains test structures for serial debugging purposes, so a small levitator can be attached and controlled.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

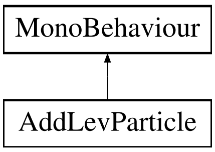
Chapter 4

Class Documentation

4.1 AddLevParticle Class Reference

Renders pre-game, allowing a Unity Prefab and an instantiation area for a LevParticle to be specified. The input GameObjects are provided pre-build within the Unity editor.

Inheritance diagram for AddLevParticle:



Public Member Functions

- void **CreateParticle** ()

  Creates a new GameObject with the levParticlePrefab in the instArea.

Public Attributes

- GameObject **levParticlePrefab**

  The GameObject to be used for the LevParticle. Provided pre-build as a prefab.
- GameObject **instArea**

  The GameObject to locate the instantiation area. Provided as an empty axis anywhere within the space, ideally close to the levitator.

4.1.1 Detailed Description

Renders pre-game, allowing a Unity Prefab and an instantiation area for a LevParticle to be specified. The input GameObjects are provided pre-build within the Unity editor.

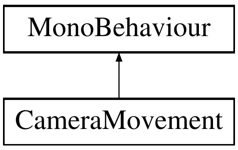
The documentation for this class was generated from the following file:

- AddLevParticle.cs

4.2 CameraMovement Class Reference

Controls the zoom level by physically moving the camera within the space. The camera moves towards the levitator (as a fixed point). This class is attached to the HexCam camera within Unity.

Inheritance diagram for CameraMovement:



Public Attributes

- Transform **DummyObject**

  GameObject Transform to act as an anchor point for the camera. A dummy object for the centre of the levitator.
- float **positionSpeed** = 0.1F

  Specifies the speed at which the camera moves. Provided in the Unity editor.
- int **maxZoom** = 90

  Specifies how close the camera can move towards the dummy (levitator centre).
- int **minZoom** = 23

  Specifies how far the camera can move away from the dummy (levitator centre).

Private Member Functions

- void **Start** ()
- void **Update** ()

Private Attributes

- Vector3 **CameraPosition**

  Stores the camera position as 3D-coordinates.

4.2.1 Detailed Description

Controls the zoom level by physically moving the camera within the space. The camera moves towards the levitator (as a fixed point). This class is attached to the HexCam camera within Unity.

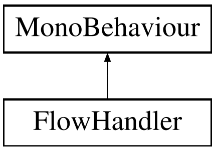
The documentation for this class was generated from the following file:

- CameraMovement.cs

4.3 FlowHandler Class Reference

Manages the workflow of the program. Graphic Raycasters are used to determine interaction with UI elements such as menus and buttons, as well selecting and moving particles.

Inheritance diagram for FlowHandler:



Public Attributes

- new Camera **camera**

    Camera reference to main HexCam. Provided within the Unity editor.
- GameObject **uiCanvas**

    UI Canvas element for buttons and text. Provided within the Unity editor.
- Material **normalMaterial**

    Material asset for specifying the visual properties of a normal unselected LevParticle. Provided within the Unity editor.
- Material **selectedMaterial**

    Material asset for specifying the visual properties of the selected LevParticle. Provided within the Unity editor.

Private Member Functions

- void **Start** ()
- void **Update** ()

Private Attributes

- GraphicRaycaster **m_Raycaster**

    GraphicRaycaster element of the uiCanvas. Used to detect user hits on graphic elements.
- EventSystem **m_EventSystem**

    Stores interaction elements as events. Provided within the Unity Editor.
- PointerEventData **m_PointerEventData**

    Stores mouse pointer information.
- GameObject **selected**

    References the currently selected particle as a Gameobject.
- LevParticle **SelectedLevParticle**

    References the currently selected particle as a LevParticle.
- Renderer **SelectedRenderer**

    Holds the Renderer properties of the selected LevParticle.
- bool **isSelected**

    Keeps track of whether a LevParticle is selected or not.
- int **trLayer**

    Specifies the transducer layer as set in Unity. This layer is excluded from the raycaster.
- Vector3 **spoint**

    Coordinates of the start point for the selected LevParticle's trajectory.
- Vector3 **epoint**

    Coordinates of the end point for the selected LevParticle's trajectory.
- bool **isCreatingTraj**

    Keeps track of whether a trajectory is being created or not.

4.3.1 Detailed Description

Manages the workflow of the program. Graphic Raycasters are used to determine interaction with UI elements such as menus and buttons, as well selecting and moving particles.

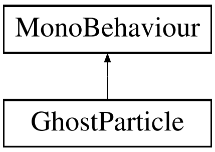
The documentation for this class was generated from the following file:

- FlowHandler.cs

4.4 GhostParticle Class Reference

A ghost particle for keeping track of a LevParticle's trajectory. Faded and slighlty transparent partices placed along points of a trajectory to indicate future positions of particles. Useful in checking for path collisions and visualising the workspace.

Inheritance diagram for GhostParticle:



Public Member Functions

- Vector3 GetPostion ()

  Gets the position of the ghost particle.
- Vector2 GetXYPositionRounded ()

  Gets the rounded 2D coordinates of the ghost particle. The z-coordinate is used as the y-coordinate, as in the Unity space.
- Vector2 GetXYPosition ()

  Gets the unrounded 2D coordinates of the ghost particle. The z-coordinate is used as the y-coordinate, as in the Unity space.
- List< Transducer > FindNearbyTransducers ()

  Finds Transducers in the vicinity of the ghost particle. The position of transducers and ghost particles are projected onto a 2D plane. This is done by checking collisions between the SphereCollider and elongated colliders of the Transducers. The closeness is defined by the SphereCollider radius.

Private Member Functions

- void **Awake** ()
- void **OnValidate** ()
- void **Start** ()
- void **Update** ()

Private Attributes

- Vector3 **particlePos**

  Stores the coordinates of the ghost.
- List< Transducer > **NearbyTransducers**

  Stores a list of nearby Transducers to be used in controlling the movement of a LevParticle when it reaches this GhostParticle.
- float **ColliderRadius**

  Specifies the radius of the sphere collider used to determine nearby Transducers. Provided by the SphereCollider in the Unity editor.

4.4.1 Detailed Description

A ghost particle for keeping track of a LevParticle's trajectory. Faded and slighlty transparent partices placed along points of a trajectory to indicate future positions of particles. Useful in checking for path collisions and visualising the workspace.

4.4.2 Member Function Documentation

4.4.2.1 FindNearbyTransducers()

```
List< Transducer > GhostParticle.FindNearbyTransducers ( )
```

Finds Transducers in the vicinity of the ghost particle. The position of transducers and ghost particles are projected onto a 2D plane. This is done by checking collisions between the SphereCollider and elongated colliders of the Transducers. The closeness is defined by the SphereCollider radius.

Returns

A list of nearby Transducer objects

4.4.2.2 GetPostion()

```
Vector3 GhostParticle.GetPostion ( )
```

Gets the position of the ghost particle.

Returns

Position of the ghost particle

4.4.2.3 GetXYPosition()

```
Vector2 GhostParticle.GetXYPosition ( )
```

Gets the unrounded 2D coordinates of the ghost particle. The z-coordinate is used as the y-coordinate, as in the Unity space.

Returns

(x,y) vector coordinates of the ghost particle

4.4.2.4 GetXYPositionRounded()

```
Vector2 GhostParticle.GetXYPositionRounded ( )
```

Gets the rounded 2D coordinates of the ghost particle. The z-coordinate is used as the y-coordinate, as in the Unity space.

Returns

(x,y) rounded vector coordinates of the ghost particle

The documentation for this class was generated from the following file:

- GhostParticle.cs

4.5 GhostTransducerPositionData Class Reference

This class contains data for a given transducer along a path and the corresponding 'previous' and 'next' ghost particles The angle of exit/approach and magnitude stored in this class is used to determine the amplitude and phase control of surrounding transducers.

Public Member Functions

- **GhostTransducerPositionData** (Transducer tr, GhostParticle gs1, GhostParticle gs2)
- float GetDist ()

  Returns the 'next' distance.
- void **DivideDist** (float d)

Public Attributes

- readonly float **ang**
- readonly Transducer **trs**
- readonly GhostParticle **gst1**
- readonly GhostParticle **gst2**

Private Attributes

- float **dist**

4.5.1 Detailed Description

This class contains data for a given transducer along a path and the corresponding 'previous' and 'next' ghost particles The angle of exit/approach and magnitude stored in this class is used to determine the amplitude and phase control of surrounding transducers.

4.5.2 Member Function Documentation

4.5.2.1 GetDist()

```
float GhostTransducerPositionData.GetDist ( )
```

Returns the 'next' distance.

Returns

The distance beween the transducer and the next ghost particle

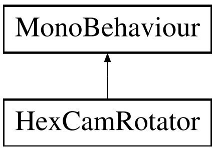
The documentation for this class was generated from the following file:

- LevParticle.cs

4.6 HexCamRotator Class Reference

Controls the camera angle by physically rotating the camera about the centre of the levitator.

Inheritance diagram for HexCamRotator:



Public Attributes

- float **rotationSpeed** = 1

  Specifies how quickly the camera can rotate.

Private Member Functions

- void **Start** ()
- void **Update** ()

4.6.1 Detailed Description

Controls the camera angle by physically rotating the camera about the centre of the levitator.

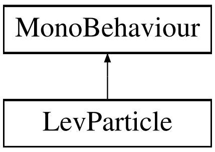
The documentation for this class was generated from the following file:

- HexCamRotator.cs

4.7 LevParticle Class Reference

A LevParticle is a particle to be levitated within the space. This class is attached to a particle by AddLevParticle, which is controlled by the FlowHandler. All the information concerning a particle is stored in this class; trajectory, ghosts, postion, status etc.

Inheritance diagram for LevParticle:



Public Member Functions

- Vector3 GetPosition ()

  Gets the postion of the particle.
- void SetSelect (bool sel)

  Sets the select status of the particle.
- void MoveX (int dir)

  Moves the particle in the x-direction.
- void MoveY (int dir)

  Moves the particle in the y-direction.
- void MoveZ (int dir)

  Moves the particle in the z-direction.
- void **DeleteParticle** ()

  Deletes the particle. This also removes the ghost particles.
- void AddTrajectory (Vector3 A, Vector3 B)

  Creates and adds a new Trajectory to the list of the particle's Trajectory. The startpoint of the trajectory must be the same as the endpoint of the previous trajectory (if it exists).
- List<(List< GhostTransducerPositionData >, List< GhostTransducerPositionData >)> GetFullTrajectoryTransducerDataList ()

  Gets the full trajectory-transducer data list.

Public Attributes

- GameObject **ghostParticlePrefab**

  Prefab indicating the GhostParticles along trajectories. Provided within the Unity editor.

Private Member Functions

- void **Awake** ()
- void **Start** ()
- void **Update** ()
- void **CreateGhostParticles** ()

  Creates GhostParticles along the particle's trajectory. The ghost parent is deleted and recreated. To this effect, all the previous ghost particles are deleted and created from the trajectories list. This function is called by the AddTrajectory function.

Private Attributes

- Vector3 **particlePos**

  Stores the position of the particle.
- bool **selected**

  Keeps track of whether the particle is selected or not.
- List< Trajectory > **Trajectories**

  Stores a list of Trajectory objects. Multiple trajectories can be chained.
- GameObject **ghostParent**

  Gameobject which holds the GhostParticles of the particle. Each LevParticle keeps track of its ghosts.
- List<(List< GhostTransducerPositionData >, List< GhostTransducerPositionData >)> **FullTrajectory←↩ TransducerDataList**

  Stores a list of tuples which contains a pair of data. The first element of the tuple describes the relationship between a target position and a current transducer. The second element of the tuple describes the relationship between a target position and the next transducer. The list of these tuples describes the full trajectory in terms of target positions and involved transducers, step-by-step.

4.7.1 Detailed Description

A LevParticle is a particle to be levitated within the space. This class is attached to a particle by AddLevParticle, which is controlled by the FlowHandler. All the information concerning a particle is stored in this class; trajectory, ghosts, postion, status etc.

4.7.2 Member Function Documentation

4.7.2.1 AddTrajectory()

```
void LevParticle.AddTrajectory (
            Vector3 A,
            Vector3 B )
```

Creates and adds a new Trajectory to the list of the particle's Trajectory. The startpoint of the trajectory must be the same as the endpoint of the previous trajectory (if it exists).

Parameters

A	3D-coordinates indicating the start point of the trajectory
B	3D-coordinates indicating the end point of the trajectory

4.7.2.2 GetFullTrajectoryTransducerDataList()

```
List<(List< GhostTransducerPositionData >, List< GhostTransducerPositionData >)> LevParticle.←↩
GetFullTrajectoryTransducerDataList ( )
```

Gets the full trajectory-transducer data list.

Returns

List of the trajectory-transducer data list

4.7.2.3 GetPosition()

```
Vector3 LevParticle.GetPosition ( )
```

Gets the postion of the particle.

Returns

3D coordinates of the particle

4.7.2.4 MoveX()

```
void LevParticle.MoveX (
             int dir )
```

Moves the particle in the x-direction.

Parameters

dir	Specifies the direction. -1 indicates backwards and 1 indicates forward.

4.7.2.5 MoveY()

```
void LevParticle.MoveY (
             int dir )
```

Moves the particle in the y-direction.

Parameters

dir	Specifies the direction. -1 indicates backwards and 1 indicates forward.

4.7.2.6 MoveZ()

```
void LevParticle.MoveZ (
             int dir )
```

Moves the particle in the z-direction.

Parameters

dir	Specifies the direction. -1 indicates backwards and 1 indicates forward.

4.7.2.7 SetSelect()

```
void LevParticle.SetSelect (
            bool sel )
```

Sets the select status of the particle.

Parameters

sel	The particle is selected?

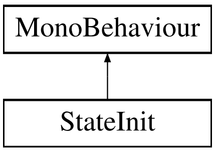
The documentation for this class was generated from the following file:

- LevParticle.cs

4.8 StateInit Class Reference

Initialises the program state: handles external interfacing and control algorithms.

Inheritance diagram for StateInit:



Public Member Functions

- void **InitializeArrays** ()

  Adds transducers to their respective plate arrays. The transducers are pre-labelled in Unity Editor.
- bool UpdateLevState ()

  Updates the state of the physical levitator. Interfaces with an arduino TODO: (Dependency) Implement updator for large array when PCB and Transducers are available Current placeholde uses debug values.
- int **ConvertGTPDtoPhaseAmplitude** (GhostTransducerPositionData gtpd, bool far)
- void **OnApplicationQuit** ()

Public Attributes

- Transducer

  Converts GhostTransducerPositionData to phase and amplitude. The current placeholder is a crude implementation which switches phases and amplitudes from low to high based on distances and angles. The positions availabe are Triangle (T, 3 transducers), Between (B, 2 transducers), Center (C, 1 transducer) TODO: (Dependency) Implement solver for large array when PCB and Transducers are available.
- **int**

Private Member Functions

- void **Awake** ()
- void **Start** ()
- void **Update** ()
- List< List< List<(Transducer, int, int)> > > CalculateStateChange ()

  Calculates the state change for needed to advance all particles per timestep. The combined PAT (Phase-Array-↩ Time) list is returned. The CombinedPATList represents a combination of the Phase and Amplitude formats of each trajectory for each particle The 'Time' aspect is introduced when each movement of each particle is aligned. This indicates that Move 1 of every particle occurs in the same step, then Move 2 of every particle occurs next...

Private Attributes

- SerialPort **ArduinoSerial**

  Serial Port for arduino. Will differ based on ports and OS.
- List< Transducer > **BottArray**

  Array of transducers for bottom plate.
- List< Transducer > **TopArray**

  Array of transducers for top plate.
- float **HexCntr_z**

  Z-axis value of the levitator centre.

4.8.1 Detailed Description

Initialises the program state: handles external interfacing and control algorithms.

4.8.2 Member Function Documentation

4.8.2.1 CalculateStateChange()

```
List< List< List<(Transducer, int, int)> > > StateInit.CalculateStateChange ( )  [private]
```

Calculates the state change for needed to advance all particles per timestep. The combined PAT (Phase-Array-↩ Time) list is returned. The CombinedPATList represents a combination of the Phase and Amplidute formats of each trajectory for each particle The 'Time' aspect is introduced when each movement of each particle is aligned. This indicates that Move 1 of every particle occurs in the same step, then Move 2 of every particle occurs next...

Returns

A list of lists containing lists of 3-item tuples. (Transducer, Phase, Amplitude)

4.8.2.2 UpdateLevState()

```
bool StateInit.UpdateLevState ( )
```

Updates the state of the physical levitator. Interfaces with an arduino TODO: (Dependency) Implement updator for large array when PCB and Transducers are available Current placeholde uses debug values.

Returns

4.8.3 Member Data Documentation

4.8.3.1 Transducer

```
StateInit.Transducer
```

Converts GhostTransducerPositionData to phase and amplitude. The current placeholder is a crude implementation which switches phases and amplitudes from low to high based on distances and angles. The positions availabe are Triangle (T, 3 transducers), Between (B, 2 transducers), Center (C, 1 transducer) TODO: (Dependency) Implement solver for large array when PCB and Transducers are available.

Parameters

gtpd	GhostTransducerPositionData to extract data from
far	Indicates whether a transducer is within the Area-Of-Interest of a particle (currently unused)

Returns

The documentation for this class was generated from the following file:

- StateInit.cs

4.9 Trajectory Class Reference

This class contains Trajectory data related to a given LevParticle.

Public Member Functions

- **Trajectory** (Vector3 A, Vector3 B, float res)
- Vector3 GetStartPoint ()

  Gets the start point of the trajectory.
- Vector3 GetEndPoint ()

  Gets the end point of the trajectory.
- List< Vector3 > GetPath ()
- List< Vector3 > CalculatePath ()

  Calculate the path using the resolution of the trajectory. Places points inbetween the start and end points.
- void AddGhostParticle (GhostParticle gst)

  Adds a GhostParticle along the trajectory.
- List< GhostParticle > GetGhostParticles ()

  Gets the list of GhostParticles for the trajectory.
- List<(List< GhostTransducerPositionData >, List< GhostTransducerPositionData >)> GetTrajectoryTransducerData ()

  Returns data which relates points on a trajectory to transducers surrounding the trajectory. See CalculateTrajectoryTransducerData().

Private Member Functions

- void **CalculateTrajectoryTransducerData** ()

  From the nearby transducers on a given path, calculate near and far transducers to be used by the chosen phase and amplitude manipulation method. This takes into account two ghost particles in sequence, allowing for the future moves of the particle to be predicted The transducers involved for each ghost particle are separated, with the intersecting transducers being allocated to the nearer ghost Adds a tuple of lists to the TrajectoryTransducerData The first element contains GhostTransducerPositionData pertaining to 'Postion 2, Transducer 1' The second element contains GhostTransducerPositionData pertaining to 'Postion 2, Transducer 2'.
- void StandardiseTrajectoryTransducerData (List< GhostTransducerPositionData > gtpd_list, float d_max)

  Standardises (min-max) the distances in trajectory data. Useful for debugging purposes.

Private Attributes

- readonly Vector3 **StartPoint**

  3D-coordinates specifying the start point of the trajectory.
- readonly Vector3 EndPoint
- readonly List< Vector3 > **tPath**

  A list of 3D-coordinates (points) connecting the start point to the end point.
- readonly float **Res**

  The resolution of the trajectory in Unity units. Controls the frequency of intermediate points.
- List< GhostParticle > **GhostParticles**

  Stores the list of GhostParticles along the trajectory.
- List<(List< GhostTransducerPositionData >, List< GhostTransducerPositionData >)> **Trajectory↩ TransducerData**

  Stores the list of data relating the GhostParticles and Transducers along the trajectory.

4.9.1 Detailed Description

This class contains Trajectory data related to a given LevParticle.

4.9.2 Member Function Documentation

4.9.2.1 AddGhostParticle()

```
void Trajectory.AddGhostParticle (
            GhostParticle gst )
```

Adds a GhostParticle along the trajectory.

Parameters

gst	GhostParticle to be added

4.9.2.2 CalculatePath()

```
List< Vector3 > Trajectory.CalculatePath ( )
```

Calculate the path using the resolution of the trajectory. Places points inbetween the start and end points.

Returns

List of all the points along the trajectory

4.9.2.3 GetEndPoint()

```
Vector3 Trajectory.GetEndPoint ( )
```

Gets the end point of the trajectory.

Returns

3D-Coordinates of the end point

4.9.2.4 GetGhostParticles()

```
List< GhostParticle > Trajectory.GetGhostParticles ( )
```

Gets the list of GhostParticles for the trajectory.

Returns

List of GhostParticles

4.9.2.5 GetPath()

```
List< Vector3 > Trajectory.GetPath ( )
```

Gets the path of the trajectory.

Returns

List containing 3D-Coordinates of all points along the trajectory

4.9.2.6 GetStartPoint()

```
Vector3 Trajectory.GetStartPoint ( )
```

Gets the start point of the trajectory.

Returns

3D-Coordinates of the start point

4.9.2.7 GetTrajectoryTransducerData()

```
List<(List< GhostTransducerPositionData >, List< GhostTransducerPositionData >)> Trajectory.↩
GetTrajectoryTransducerData ( )
```

Returns data which relates points on a trajectory to transducers surrounding the trajectory. See CalculateTrajectoryTransducerData().

Returns

A list of tuples containing a pair of lists (GhostTransducerPositionData)

4.9.2.8 StandardiseTrajectoryTransducerData()

```
void Trajectory.StandardiseTrajectoryTransducerData (
            List< GhostTransducerPositionData > gtpd_list,
            float d_max ) [private]
```

Standardises (min-max) the distances in trajectory data. Useful for debugging purposes.

Parameters

gtpd_list	GhostTransducerPositionData list on trajectory
d_max	Maximum distance on trajectory

4.9.3 Member Data Documentation

4.9.3.1 EndPoint

```
readonly Vector3 Trajectory.EndPoint  [private]
```

3D-coordinates specifying the end point of the trajectory.

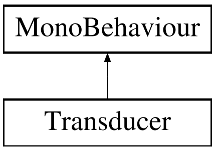
The documentation for this class was generated from the following file:

- LevParticle.cs

4.10 Transducer Class Reference

Class for storing transducer data.

Inheritance diagram for Transducer:



Public Member Functions

- void Init (int arr, int ind, float bCenter_z)
  Initialises the transducer by assigning an array (plate) and index to it.
- void **Activate** ()
  Turns the transducer on.
- void **Deactivate** ()
  Turns the transducer off.
- bool IsActive ()
  Gets the state of the transducer; on or off.
- Vector3 GetPosition ()
  Gets the XYZ postion of the transducer.
- Vector2 GetXYPositionRounded ()
  Gets the XY postion of the transducer rounded to the nearest integer.
- Vector2 GetXYPosition ()
  Gets the XY postion of the transducer.
- int GetPhase ()
  Returns the phase of the transducer.
- int GetAmplitude ()
  Returns the amplitude of the transducer.
- void SetPhase (int p)
  Sets the phase of the transducer.
- void SetAmplitude (int a)
  Sets the amplitude of the transducer.

Private Member Functions

- void **Start** ()
- void **Update** ()

Private Attributes

- Vector3 **tPosition**
- int **tIndex**
- int **tArr**
- int **tPhase**
- int **tAmplitude**
- bool **used**

4.10.1 Detailed Description

Class for storing transducer data.

4.10.2 Member Function Documentation

4.10.2.1 GetAmplitude()

```
int Transducer.GetAmplitude ( )
```

Returns the amplitude of the transducer.

Returns

Currently, Amplitude as a percentage

4.10.2.2 GetPhase()

```
int Transducer.GetPhase ( )
```

Returns the phase of the transducer.

Returns

Currently, Phase as a percentage

4.10.2.3 GetPosition()

```
Vector3 Transducer.GetPosition ( )
```

Gets the XYZ postion of the transducer.

Returns

3D position vector

4.10.2.4 GetXYPosition()

```
Vector2 Transducer.GetXYPosition ( )
```

Gets the XY postion of the transducer.

Returns

2D position vector

4.10.2.5 GetXYPositionRounded()

```
Vector2 Transducer.GetXYPositionRounded ( )
```

Gets the XY postion of the transducer rounded to the nearest integer.

Returns

2D position vector

4.10.2.6 Init()

```
void Transducer.Init (
             int arr,
             int ind,
             float bCenter_z )
```

Initialises the transducer by assigning an array (plate) and index to it.

Parameters

arr	The parent plate, either top or bottom

<param name="ind" The index of the transducer, to match any physical implementation>

Parameters

bCenter_z	Specifies the center of the transducer collider

4.10.2.7 IsActive()

```
bool Transducer.IsActive ( )
```

Gets the state of the transducer; on or off.

Returns

True if transducer is on

4.10.2.8 SetAmplitude()

```
void Transducer.SetAmplitude (
             int a )
```

Sets the amplitude of the transducer.

Parameters

a	Amplitude (currently 0%-100%)

4.10.2.9 SetPhase()

```
void Transducer.SetPhase (
             int p )
```

Sets the phase of the transducer.

Parameters

p	Phase (currently 0%-100%)

The documentation for this class was generated from the following file:

- Transducer.cs

Index