EXAMPLES IN THE PACKAGE!

Thank you for purchase!

Description :

With this asset you can count mechanic, math, physic and material science formulas and calculations.

Using :

If you want to use it, just call SmartMath namespace in the script :

```
using SmartMath;
```

Possibilities :

- **maths**

-percentage(float):

With this method you can calculate the percentage of numbers.

***(Parameters:** number : float , percent : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.Percentage (100, 6);

    }
}
```

-per mille(float):

With this method you can calculate the per mille of numbers.

***(Parameters:** number : float , permill : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.PerMille (100, 30);

    }
}
```

-average(double):

With this method you can calculate the average of numbers.

***(Parameters:** i : double[ ])*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.Average (3,5,6,3,1,3);

    }
}
```

-max(double):

With this method you can calculate the maximum of numbers.

***(Parameters:** i : double[ ])*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.Max (3, 6, 3, 11);

    }
}
```

-min(double):

With this method you can calculate the minimum of numbers.

***(Parameters:** i : double[ ])*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.Min (3, 6, 3, 11);

    }
}
```

-sum(double):

With this method you can calculate the sum of numbers.

***(Parameters:** i : double[ ])*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.Sum (5, 3, 6, 7, 7);

    }
}
```

-nroot(double):

With this method you can calculate the given root of numbers.

***(Parameters:** root : double, number: double)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.NRoot (5, 15);

    }
}
```

-triangle perimeter(float):

With this method you can calculate the perimeter of a triangle.

***(Parameters:** a : float, b: float, c: float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.TrianglePerimeter (5, 3, 2);

    }
}
```

-square perimeter(float):

With this method you can calculate the perimeter of a square.

***(Parameters:** a: float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.SquarePerimeter (3);

    }
}
```

-rhombus perimeter(float):

With this method you can calculate the perimeter of a rhombus.

***(Parameters:** a: float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.RhombusPerimeter (5);

    }
}
```

-circle perimeter(float):

With this method you can calculate the perimeter of a circle.

***(Parameters:** r: float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.CirclePerimeter (5);

    }
}
```

-law of sines(double):

This method represent the law of sines.

***(Parameters:** side1 : float, angle1 : float, angle2 : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.SinesLaw (5, 73, 62);

    }
}
```

-law of cosines(double):

This method represent the law of cosines.

***(Parameters:** side1 : float, side2 : float, angle1 : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.CosineLaw (6, 2, 60);

    }
}
```

-Pythagoras theorem(double):

With this method you can calculate a side of a triangle.

***(Parameters:** side1 : float, side2 : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Math.Pythagoras (5, 4);

    }
}
```

-quadratic equation(minus) double:

With this method you can calculate the quadratic equation of numbers(the first solution).

***(Parameters:** a : float, b : float, c : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = SmartMath.Math.QuadraticEquationMinus (2, -33, 115);
    }
}
```

-quadratic equation(plus) double:

With this method you can calculate the quadratic equation of numbers(the second solution).

***(Parameters:** a : float, b : float, c : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = SmartMath.Math.QuadraticEquationPlus (2, -33, 115);
    }
}
```

-Geometric Sequence(double):

With this method you can calculate the value of a geometric sequence's nth element.

(Parameters: r : float, a : float, n: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update()
    {

        result = Math.GeometricSequence (2, 5, 10);

    }
```

-Geometric Sequence Sum(double):

With this method you can calculate the sum of the first n element of a geometric sequence.

(Parameters: r : float, a : float, n: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update()
    {

        result = Math.GeometricSequenceSum (2, 5, 10);

    }
```

-Arithmetic Sequence(double):

With this method you can calculate the value of a arithmetic sequence's nth element.

(Parameters: d : float, a : float, n: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update()
    {

        result = Math.ArithmeticSequence (2, 5, 10);

    }
```

-Arithmetic Sequence Sum(double):

with 2 formulas!!

With this method you can calculate the sum of the first n element of a arithmetic sequence.

1,(Parameters: a1 : float, an: float, n: float)

2,(Parameters: *a1* : float, n : float, d: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update()
    {

        result = Math.ArithmeticSequenceSum1 (2, 5, 10);

    }
```

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update()
    {

        result = Math.ArithmeticSequenceSum2 (2, 5, 10);

    }
```

-Compound Interest(double):

With this method you can calculate the future value of an amount(for example money).

(Parameters: P : float, i : float, n : float, t : float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update()
    {

        result = Math.CompoundInterest (5000, 0.05, 12, 10);

    }
```

-Octahedron:

With this method you can calculate the area and the volume of an octahedron.
(**parameters**: a: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;

    void Update ()
    {
        result1 = Math.OctahedronArea(5);
        result2 = Math.OctahedronVolume(5);

    }
}
```

-Dodecahedron:

With this method you can calculate the area and the volume of a dodecahedron.
(**parameters**: a: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;

    void Update ()
    {
        result1 = Math.DodecahedronArea(2);
        result2 = Math.DodecahedronVolume(2);

    }
}
```

-Cube:

With this method you can calculate the area and the volume of a cube.
(**parameters**: a: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;

    void Update ()
    {
        result1 = Math.CubeArea(6);
        result2 = Math.CubeVolume(6);

    }
}
```

-Cuboid:

With this method you can calculate the area and the volume of a cuboid.
(**parameters**: a: float, b:float, c:float, d:float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;

    void Update ()
    {
        result1 = Math.CuboidArea(1,3,5,0.1f);
        result2 = Math.CuboidVolume(1,3,5);

    }
}
```

-Cylinder:

With this method you can calculate the area and the volume of a cylinder.

(**parameters**: a: float h: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;

    void Update ()
    {
        result1 = Math.CylinderArea(1.5f, 2.2f);
        result2 = Math.CylinderVolume(1.5f, 2.2f);

    }
}
```

-Cone:

With this method you can calculate the area and the volume of a cone.

(**parameters:** r: float, a: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;

    void Update ()
    {
        result1 = Math.ConeArea(2, 3.3f);
        result2 = Math.ConeVolume(2, 3.3f);

    }
}
```

-Sphere:

With this method you can calculate the area and the volume of a sphere.

(**parameters:** R: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;

    void Update ()
    {
        result1 = Math.SphereArea(5.5f);
        result2 = Math.SphereVolume(5.5f);

    }
}
```

-Ellipsoid:

With this method you can calculate the volume of an ellipsoid.
(**parameters:** a: float, b:float, c:float, R: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update ()
    {
        result = Math.EllipsoidVolume(2, 4, 3);

    }
}
```

-Tetrahedron:

With this method you can calculate the area and the volume of a tetrahedron.
(**parameters**: a: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;

    void Update ()
    {
        result1 = Math.TetrahedronArea(2.25f);
        result2 = Math.TetrahedronVolume(2.25f);

    }
}
```

-Icosahedron:

With this method you can calculate the area and the volume of an icosahedron.
(**parameters**: a: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;

    void Update ()
    {
        result1 = Math.IcosahedronArea(3.33f);
        result2 = Math.IcosahedronVolume(3.33f);

    }
}
```

-Distance 3D:

With this method you can calculate the distance of 2 objects on x,y and z axis.
(**parameters**: vector3 firstpoint, vector3 secondpoint)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update ()
    {
        result = Math.Distance3D(new Vector3(120.3f, 43.4f, 55.5f), new Vector3(550, 60f, 55.5f));
    }
}
```

-Distance 2D:

With this method you can calculate the distance of 2 objects on x and y axis.
(**parameters**: firstpoint: float, secondpoint: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update ()
    {
        result = Math.Distance2D(new Vector2(120.3f, 43.4f), new Vector2(550, 60f));
    }
}
```

-Distance 1D:

With this method you can calculate the distance of 2 objects on x axis.
(**parameters**: vector1 firstpoint, vector1 secondpoint)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update ()
    {
        result = Math.Distance1D(525.27f, 123.96f);
    }
}
```

-Remarkable Identities(3):

With this method you can calculate the remarkable identities.
Identity1: $(a+b)^2$
Identity2: $(a-b)^2$
Identity3: `(a+b)*(a-b)`
(**parameters**: a: float, b: float)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result1;
    public double result2;
    public double result3;

    void Update ()
    {
        result1 = Math.RemarkableIdentities1(2.3f, 4.3f);
        result2 = Math.RemarkableIdentities2(2.3f, 4.3f);
        result3 = Math.RemarkableIdentities3(2.3f, 4.3f);
    }
}
```

-Factorial

With this function you can calculate the factorial of a number

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public int result;


    void Update ()
    {

        result = Math.Factorial(5);

    }
}
```

2. Physic

-Pendulum Gravity Acceleration(double):

With this method you can calculate the gravity acceleration of a pendulum.

(Parameters: *l : double, T : double)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Physic.PendulumGravityAcceleration (0.623,1.6374);

    }
}
```

-Frequency

With this method you can the frequency.

(Parameters: *T : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update()
    {

        result = Physic.Frequency (10);

    }
```

-Ohm's Law

***(Parameters:** V : float, R : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update()
    {

        result = Physic.OhmLaw (6, 8);

    }
```

-Periodic Wave

With this method you can the periodic wave.

***(Parameters:** lambda : float, f : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update()
    {

        result = Physic.PeriodicWave (5, 10);

    }
```

-Coulomb's law

With this function you can calculate Coulomb's law

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.Coulomb (5, 4, 3, 2);

    }
}
```

-Capacitance

With this function you can calculate the capacitance

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.Capacitance (2, 1.04f);

    }
}
```

-Sensible heat

With this function you can calculate the sensible heat

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.SensibleHeat (6, 5, 7);

    }
}
```

-Latent heat

With this function you can calculate the latent heat

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.LatentHeat (5, 3.3f);

    }
}
```

-Thermodynamics first law

With this function you can calculate the thermodynamics first law

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.ThermoDinFirstLaw (8, 3.01f);

    }
}
```

-Internal energy

With this function you can calculate the internal energy

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.InternalEnergy (5, 2.2f, 4);

    }
}
```

-Mach number

With this function you can calculate the mach number

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.MachNumber (2, 2.7f);

    }
}
```

-Index of refraction

With this function you can calculate the index of refraction

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.IndexOfRefraction (6, 5);

    }
}
```

-Mass energy

With this function you can calculate the mass energy

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.MassEnergy (2, 3);

    }
}
```

-Photon energy

With this function you can calculate the photon energy

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.PhotonEnergy (5, 3);

    }
}
```

-Photon momentum

With this function you can calculate the photon momentum

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.PhotonMomentum (4, 2.1f);

    }
}
```

-Photoelectric effect

With this function you can calculate the photoelectric effect

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Physic.PhotoelectricEffect (2, 1, 2.05f);

    }
}
```

3. Mechanic

-center of gravity(x & y):

x(double)

With this method you can calculate the x coordinate of a body's gravity center.

***(Parameters:** x1 : double, a1 : double, x2 : double, a2 : double)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Mechanic.CenterOfGravityX (10, 1200, 30, 1200);

    }
}
```

y(double)

With this method you can calculate the y coordinate of a body's gravity center.

***(Parameters:** y1 : double, a1 : double, y2 : double, a2 : double)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = Mechanic.CenterOfGravityY (50, 1200, 10, 1200);

    }
}
```

-Centripetal accel.

With this function you can calculate the centripetal acceleration

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.CentripetalAcceleration (2, 6);

    }
}
```

-Weight

With this function you can calculate the weight

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.Weight (2, 9.5f);

    }
}
```

-Average speed

With this function you can calculate the average speed

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.AverageSpeed (6, 2);

    }
}
```

-Distance

With this function you can calculate the distance

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.Distance (4, 11);

    }
}
```

-Average accel.

With this function you can calculate the average acceleration

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.AverageAcceleration (1.1f, 2);

    }
}
```

-Kinetic energy

With this function you can calculate the kinetic energy

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.KineticEnergy (4, 2.2f);

    }
}
```

-Mechanical efficie.

With this function you can calculate the mechanical efficiency

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.MechanicalEfficiency (5, 5);

    }
}
```

-Average power

With this function you can calculate the average power

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.AveragePower (2, 2);

    }
}
```

-Torque

With this function you can calculate the torque

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.Torque (4, 2, 6);

    }
}
```

-Newton’s 2. law

With this function you can calculate Newton’s second law.

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.Newton2 (7, 5);

    }
}
```

-Impulse moment.

With this function you can calculate the impulse momentum

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.ImpulseMomentum (1, 4.3f);

    }
}
```

-Hooke’s law

With this function you can calculate Hooke’s law

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.HookeLaw (2, 1.1f);

    }
}
```

-Pressure

With this function you can calculate the pressure

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.Pressure (6, 4);

    }
}
```

-Buoyancy

With this function you can calculate the buoyancy

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.Buoyancy (1, 1.1f, 1.5f);

    }
}
```

-Kinematic viscos.

With this function you can calculate the kinematic viscosity

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.KinematicViscosity (2, 2.2f);

    }
}
```

-Aerodynamic drag

With this function you can calculate the aerodynamic drag

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public float result;


    void Update ()
    {

        result = Mechanic.AerodynamicDrag (1, 2, 5, 3);

    }
}
```

4. Material science

-cutting split(float):

With this method you can calculate the cutting split.

(Parameters: *m : float, s : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = MaterialsScience.CuttingSplit (6, 3);

    }
}
```

5. Technical drawing -

tolerance(2 modes) double:

With this method you can calculate the tolerance zone of a body.

(Parameters: *i : double, q : double)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = SmartMath.TechnicalDrawing.Tolerance1 (6, 3);

    }
}
```

***(Parameters:** d1 : double, d2 : double)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = SmartMath.TechnicalDrawing.Tolerance2 (8, 5);

    }
}
```

6. General

-daily caloric needs(double):

With this method you can calculate the daily calorie need.

***(Parameters:** age : float, gender : string, weightinkg : float)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = General.Calorie (20, "male", 78);

    }
}
```

-BMI(Body Mass Index) double:

With this method you can calculate the BMI.

(Parameters: *weightinkg : double, heightinm : double)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = General.BMI (63, 1.84);

    }
}
```

-BFP(Body Fat Percentage) double:

With this method you can calculate the BFP.

(Parameters: *weightinkg : double, heightinkg : double, gender : string, age : double)*

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Example : MonoBehaviour {

    public double result;

    void Update() {

        result = General.BFP (63, 1.84, "male", 17);

    }
}
```

7. Chemie

-Periodic Table

You can find the full periodic table in this method.
(**parameters**: atomicnumber: int)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class PeriodicTable : MonoBehaviour {

    public string name;
    public int atomicnumber;
    public string mass;
    public string symbol;

    void Update ()
    {
        name = Chemie.PeriodicTable.Name(atomicnumber);
        mass = Chemie.PeriodicTable.Atomicmass(atomicnumber);
        symbol = Chemie.PeriodicTable.Symbol(atomicnumber);

    }
}
```

-Faraday Formula:

With this method you can calculate the Faraday formula.
(**parameters**: float: M, float: z, float: F, float: I, float: t)

```
using UnityEngine;
using System.Collections;
using SmartMath;

public class Examples : MonoBehaviour {

    public double result;

    void Update ()
    {
        result = Chemie.Faraday(4f, 2f, 4f, 1f, 0.4f);

    }
}
```

8. Converting

- **Temperature**

  If the SmartMath namespace is called in the script,for example you can use it like this:

  public double result;

  void Update() {

  result = Converting.Temperature.CelsiusToKelvin (30);

  }

  Types:

  -Fahrenheit

  -Kelvin

  -Celsius

- **Weight**

  If the SmartMath namespace is called in the script,for example you can use it like this:

```
public double result;
    void Update() {
        result = Converting.Weight.OuncesToPound (10);
    }
```

Types:

-Gram

-Decagram

-Kilogram

-Tonne

-Pound

-Ounces

- **Times**

If the SmartMath namespace is called in the script,for example you can it like this:

```
public double result;
    void Update() {
        result = Converting.Times.HourToMonth (21);
}
```

Types:

-Secundum

-Minute

-Hour

-Day

-Week

-Months

-Year

- **Length**

  If the SmartMath namespace is called in the script,for example you can use it like this:

```
public double result;
    void Update() {
            result = Converting.Times.Length.YardToDm (5);
}
```

Types:

-Millimeter

-Centimeter

-Meter

-Kilometer

-Mile

-Yard

-Inch

Support & suggestions:

If you have any questions, problem, or you have a suggestion for us, you can reach us on this e-mail address: **ictentertainment1@gmail.com**