Measurements Documentation

version 1.x

for Unity

Documentation: Online Documentation
Support: vrchewal.com/unity/measurements
Asset Store: http://u3d.as/16hd

Jump To:

Introduction

In most games, it helps to know certain measurements for your game objects. For example, it may help to know the width of a gameobject in inches, the height of a collider or even the distance between two objects closest edges. Unfortunately, Unity does not offer much natively without writing some code.

Measurements is a simple tool that allows you to get real time information on an object's width, height and depth for it's mesh, renderer and collider. You can also choose another gameobject and Measurements will tell you the distance between the two objects from their centers as well as their closest edges. All of this in any unit of measurement you like *(centimeters, meters, kilometers, inches, feet, miles, yards)*. Need to know the various angles between two objects? Done and done.

How To

Just add the Measurements script to your gameobject.

You can choose the *Measurement Unit* for your results. The options are: centimeters, meters, kilometers, yards, inches, feet and miles. Your results will vary depending on your *Measurement Source*. The options are Renderer, Mesh and Collider. If you wish to know information about your model, choose either Renderer or Mesh. Collider is very useful when you want to know the distance between your collider and some other gameobject.

You may also select another gameobject to be the *Distance Object*. Measurements will calculate the distance between both object's centers and closest edges, as well as angle information for their positions.

API

The following methods are open via the Measurements API.

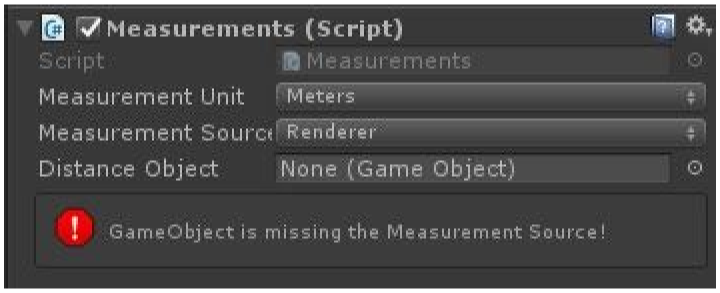
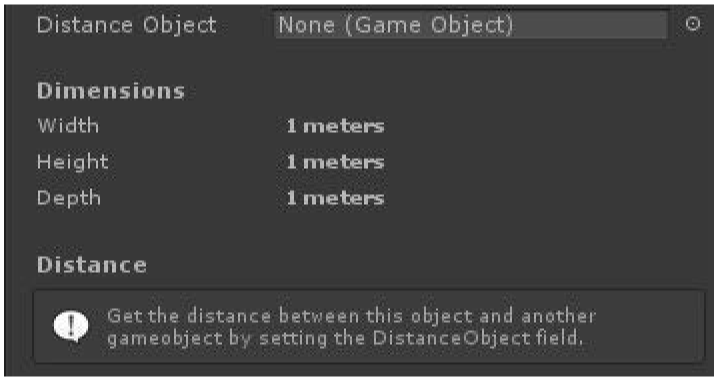
setMeasurementUnit `void`	`setMeasurementUnit(MeasurementUnits unit);` Sets the unit of measurement for your results.
getMeasurementUnit MeasurementUnits	`getMeasurementUnit();` Returns the unit of measurement for your results.
setMeasurementSource `void`	`setMeasurementSource(MeasurementSources source);` Sets the measurement source (Renderer, Mesh, Collider).
getMeasurementSource MeasurementSources	`getMeasurementSource();` Returns the measurement source (Renderer, Mesh, Collider).
setDistanceObject `void`	`setDistanceObject(GameObject obj);` Sets the distance object for distance calculations.
getDistanceObject GameObject	`getDistanceObject();` Returns the distance object used for distance calculations.
getDimensions Vector3	`getDimensions();` Returns a Vector3 with x being width, y being height and z being depth. This returns the values converted to your Measurement Unit.
getDimensionsInMeters Vector3	`getDimensionsInMeters();` Returns a Vector3 with x being width, y being height and z being depth. Returns the values in meters (Unity units).
getDistance Vector2	`getDistance();` Returns a Vector2 with x being the center to center distance and y being the edge to edge distance. Returns the values converted to your Measurement Unit.
getDistanceInMeters	`getDistanceInMeters();`

Vector2	Returns a Vector2 with x being the center to center distance and y being the edge to edge distance. Returns the values in meters (Unity units).
getConversionValue float	`getConversionValue(MeasurementUnits unit);` Returns the conversion value from meters to your selected unit. Useful for managing your own calculations.
New in 1.02	
getAngles Vector4	`getAngles();` Returns a Vector4 of degrees between the distance object and the source object. The Vector4 contains 4 values: X, Y, Z and W. You will likely only need one of these values, the trick is deciding which one. X is the horizontal (XZ) angle. This is the left/right horizontal angle, or how many degrees left or right of the source object's forward Z axis. This angle ignores the Y axis of both objects. Y is the vertical (YZ) angle. This is the up/down vertical angle, or how many degrees up or down of the source object's forward Z axis. This angle ignores the X axis of both objects. Z is the XY angle or 2D angle. This measures how many degrees from horizontal the distance object is from the source object's X axis. This is suitable for 2D scenes that do not use the Z axis. W is the Vector3 angle with all the others combined. W is the same result that Unity returns when trying to find the angle in 3D space. See the Vector3.Angle documentation for more on this.

Troubleshooting

Oops, something went wrong. You have come to the right place.

1	**Problem** You see a warning with the following: GameObject is missing the Measurement Source! **Answer** Choose a different Measurement Source or add the source that is missing. For example, if you choose Collider and the gameobject does not have a collider component, you will need to add

	the Collider before using that as a source.  Measurements (Script) Script: Measurements Measurement Unit: Meters Measurement Source: Renderer Distance Object: None (Game Object) GameObject is missing the Measurement Source!
2	**Problem** You get 0 for a width, height and depth. **Answer** Your measurement source does not have a size. Choose a different source.
3	**Problem** You see a notification for Distance with the following: Get the distance between this object and another gameobject by setting the Distance Object field. **Answer** You don't have to do anything, but if you want to know the distance information for this object and another, drag another gameobject into the Distance Object field in the inspector.  Distance Object: None (Game Object) **Dimensions** Width: 1 meters Height: 1 meters Depth: 1 meters **Distance** Get the distance between this object and another gameobject by setting the DistanceObject field.
4	**Problem** You get an angle of 90 when two objects are in the same position. **Answer** This is a feature/bug of angle calculation for Unity. You can manually check if the two objects are in the same position and decide how you want to handle it.

Support

You can get your question answered by emailing support@vrchewal.com

Thank you for your purchase of Measurements!