# IFC Importer README

IFC Importer is a plugin for importing IFC format building information models (BIM) into Unity. It preserves the IFC file's hierarchy and semantic information.

LINK to instructions video: https://youtu.be/5Slpl9_xmCU

## FEATURES

- Import IFC building information models (BIM) into Unity as GameObject hierarchies
- Works both on runtime and in the editor
- IFC models retain their property, attribute, material, quantity, types and header data
- Search game objects based on their IFC id, presentation layer and element type
- Supports both IFC 2x3 and IFC 4
- Has been tested, and confirmed to work with IFC models exported from Autodesk Revit, Archicad, Tecla Structures, Allplan and Microstation Triforma
- Works with Unity 2019.x or later on Windows, Linux and Mac
- Local process with no cloud subscription needed
- Readme with simple C# examples
- Commented C# code
- Easy to use

## INSTALLATION

1. Download the plugin from the Unity Asset Store
2. Download the latest version of IfcConvert.exe for your operating system from http://www.ifcopenshell.org/ifcconvert
3. Unzip the file you downloaded and copy the file "ifcconvert.exe" to the root of your project folder (by default to C:\UnityProjects\ProjectName)
4. This step is only needed for runtime importing. Copy the two materials from the Assets\IfcImporter\Copy_to_Resources folder to the Assets\Resources folder.

# FILES FOR TESTING

- Duplex model http://openifcmodel.cs.auckland.ac.nz/_models/030811DuplexModel-IFC-201105-05.zip
- Office model http://openifcmodel.cs.auckland.ac.nz/_models/20160414office_model_CV2_fordesign.ifc
- Garage model http://openifcmodel.cs.auckland.ac.nz/_models/20160125Autodesk_Hospital_Parking%20Garage_2015%20-%20IFC4.ifc

- Open IFC Model Repository http://openifcmodel.cs.auckland.ac.nz/Model

Email support: leo.salomaa@arcventure.fi

# USAGE

## Importing IFC models into the editor

1. Manually drag the your IFC file into the Unity Assets folder in the editor
2. The conversion starts automatically. Please wait for it to finish.
3. Place the newly created prefab into the Unity scene from the Assets folder.

NOTE: The conversion process may require a large amount of memory! If your computer runs out of memory during the process, Unity may crash.

## Importing IFC models at runtime

IFC models can be imported into Unity at runtime using IfcImporter.RuntimeImport().

### Public Static Method

IfcImporter.**RuntimeImport**( string **filepath** )

### Parameters

**filepath** string to the IFC file relative to project root. If the file is in the project root folder, the filepath is just the name of the IFC file + the .ifc extension.

### Returns

**GameObject** IFC model root. The root GameObject contains the IFC hierarchy as descendant GameObjects.

### Example

```
using UnityEngine;

public class RunTimeImport : MonoBehaviour
{
    // Imports duplex.ifc from the project root folder
    void Start()
    {
        GameObject rootObject = IfcImporter.RuntimeImport("duplex.ifc");
        rootObject.transform.Rotate(-90, 0, 0);
    }
}
```

# Using IFC semantic data in Unity

IFC GameObjects have attached components like IfcAttributes, IfcProperties, IfcMaterials, IfcQuantities, IfcTypes, IfcHeader and IfcUnits which contain lists of semantic data. For example, a wall may contain information such as its fire rating, thermal resistance, if it's load bearing, materials, length, height and so forth. The information and how it is expressed varies based on the used design software and designer preferences.

This information is stored as pairs of public lists containing pairs of information names and respective values. You can access this information as lists or using the Find() method.

## Public Methods

IfcAttributes.**Find**( string **name** )
IfcProperties.**Find**( string **name** )
IfcMaterials.**Find**( string **name** )
IfcQuantities.**Find**( string **name** )
IfcTypes.**Find**( string **name** )
IfcHeader.**Find**( string **name** )
IfcUnits.**Find**( string **name** )
IfcUnits.**Find**( string **name** )
IfcUnits.**FindSIEquivalent**( string **name** )

## Parameters

**name** of the information to find as a string.

## Returns

**string** information value of the first entry with the given name.

FindSIEquivalent() returns the divisor for converting the unit to International System of Units as a **string**. For example, if the IFC model uses feet, FindSIEquivalent("LENGTHUNIT") may return "0.3048".

### Example

```
using UnityEngine;

public class FindExample : MonoBehaviour
{
    public GameObject myIfcWall;
    void Start()
    {
        string v = myIfcWall.GetComponent<IfcAttributes>().Find("IfcElementType");
        Debug.Log( v );
    }
}
```

## Using IFC root lists

The parent of the IFC hierarchy contains parallel lists with IFC GameObjects, their IFC ids, IFC element types and IFC presentation layers. The IfcRootLists methods allow searching IFC GameObjects by these traits.

### Searching GameObjects by IFC id

IfcRootLists.**FindIfcGameObject**( string **id** )

### Parameters

**id** as a string to find in the root GameObject's hierarchy. You can view an IFC GameObject's id in its attached IfcAttributes component.

### Returns

**GameObject** corresponding to the IFC id.

### Example

```
using UnityEngine;

public class RootListExamples1 : MonoBehaviour
{
    public GameObject myIfcRoot;
    void Start()
    {
        IfcRootLists myRootLists = myIfcRoot.GetComponent<IfcRootLists>();
        string idToFind = "1xS3BCk291UvhgP2dvNtSE";
```

```
        GameObject result1 = myRootLists.FindIfcGameObject( idToFind );
        Debug.Log( result1 );
    }
}
```

## Searching IFC GameObjects by element type

IfcRootLists.**FindIfcElementTypeGameObjects**( string **elementTypeName** )

## Parameters

**elementTypeName** string to search for. For example IfcWallStandardCase, IfcDoor or IfcBeam.

## Returns

**List<GameObject>** of IFC GameObjects with the given element type attribute.

## Example

```
using System.Collections.Generic;
using UnityEngine;

public class RootListExamples2 : MonoBehaviour
{
    public GameObject myIfcRoot;
    void Start()
    {
        IfcRootLists myRootLists = myIfcRoot.GetComponent<IfcRootLists>();
        string elementTypeToFind = "IfcWallStandardCase";
        List<GameObject> elements = myRootLists.FindIfcElementTypeGameObjects(
elementTypeToFind );
        Debug.Log(elements.Count);
    }
}
```

## Searching IFC GameObjects by presentation layer

IfcRootLists.**FindIfcLayerGameObjects**( string **layerName** )

## Parameters

**layerName** string to search for. The presentation layers are defined by the design software and the designer's naming conventions.

## Returns

**List<GameObject>** of IFC GameObjects on the given IFC presentation layer.

## Example

```
using System.Collections.Generic;
using UnityEngine;

public class RootListExamples3 : MonoBehaviour
{
    public GameObject myIfcRoot;
    void Start()
    {
        IfcRootLists myRootLists = myIfcRoot.GetComponent<IfcRootLists>();
        string layerToFind = "A-DOOR";
        List<GameObject> elements = myRootLists.FindIfcLayerGameObjects(
layerToFind );
        Debug.Log(elements.Count);
    }
}
```