

Importing 3D Models

Unity 3D

- Natively supports the following formats:
 - .fbx, .obj, .dae, .3ds, .dxf
- Proprietary formats:
 - Max, Maya, Blender, Cinema4D, Modo, Lightwave, Cheetah3D, Sketchup
 - Supported if you have the software installed
- [Importing guide](#)
 - Has tips for importing each format

Textures

- Store textures in a folder called **Textures** next to the exported mesh within your unity project. This will enable Unity to find the Texture and automatically connect the Texture to the generated Material. For more information, see the [Textures](#) reference.



Pain Points

- High resolution meshes
- Textures/materials not importing



Leupold Mk4 CQ/T

High Poly



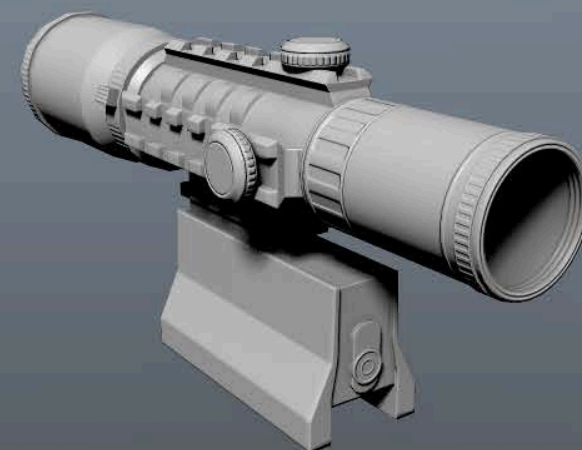
373 414 Polygons

Low Poly



1542 Polygons

Low Poly + Normal Map



1542 Polygons

Programming





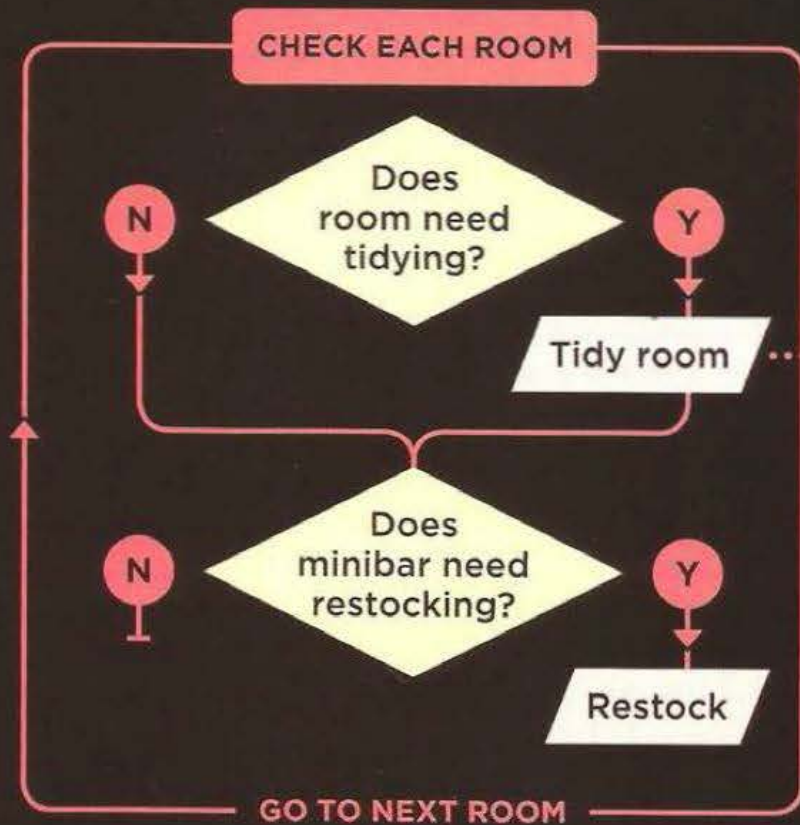
Programming?



00110001	00000000	00000000
00110001	00000001	00000001
00110011	00000001	00000010
01010001	00001011	00000010
00100010	00000010	00001000
01000011	00000001	00000000
01000001	00000001	00000001
00010000	00000010	00000000
01100010	00000000	00000000

```
int count = 0;
int sum = 0;
while (count <= 10) {
    sum += count;
    count += 1;
}
Debug.Log(sum);
```


FLOWCHART: TASKS OF A HOTEL CLEANER



LIST: STEPS REQUIRED TO TIDY A ROOM

- STEP 1** Remove used bedding
- STEP 2** Wipe all surfaces
- STEP 3** Vacuum floors
- STEP 4** Fit new bedding
- STEP 5** Remove used towels and soaps
- STEP 6** Clean toilet, bath, sink, surfaces
- STEP 7** Place new towels and soaps
- STEP 8** Wipe bathroom floor

Compilation

```
int count = 0;
int sum = 0;
while (count <= 10) {
    sum += count;
    count += 1;
}
Debug.Log(sum);
```

→
*Almost

00110001	00000000	00000000
00110001	00000001	00000001
00110011	00000001	00000010
01010001	00001011	00000010
00100010	00000010	00001000
01000011	00000001	00000000
01000001	00000001	00000001
00010000	00000010	00000000
01100010	00000000	00000000

Programming Languages in Unity



C#

UnityScript

~~Boo~~ (~2014)



C#

~~UnityScript~~ (rumored?)

~~Boo~~ (~2014)

C#
(C Sharp)





IDE Setup: VS Code



Checklist

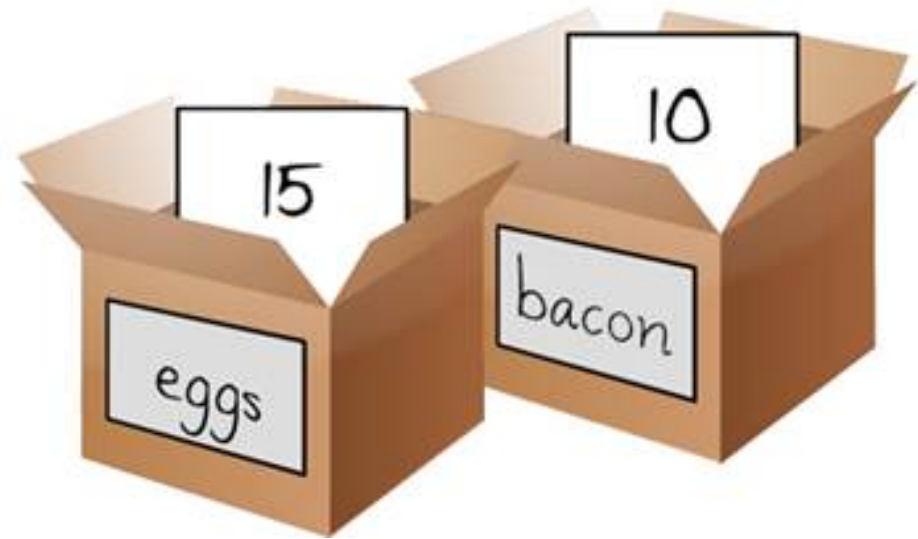
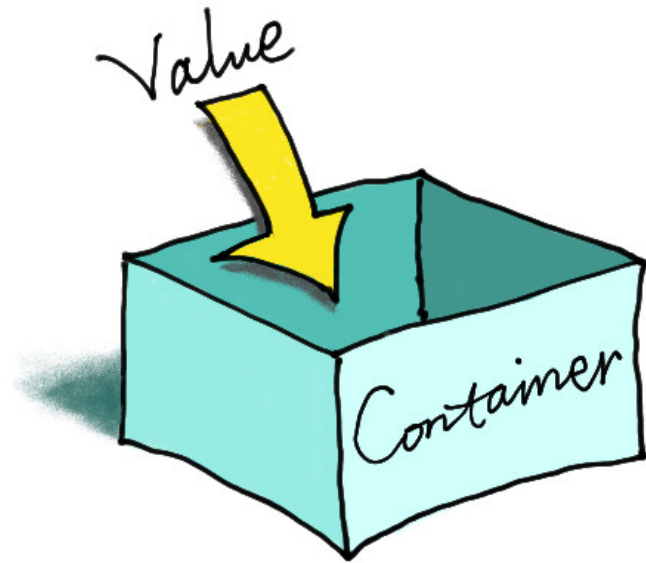
- Install [VS Code](#)
- Edit -> Preferences -> External Tools -> External Script Editor
 - Set to "Visual Studio Code"
- VS Code Extensions
 - "C#"
 - "Debugger for Unity"
 - (Optional) "Material-theme"

Logging Demo

Variables

Storing Data

Named Boxes



VARIABLE NAME

VALUE

`int numJupiterMoons = 67;`

VARIABLE TYPE

ASSIGNMENT
OPERATOR

// Camel Case

// Good – short, descriptive

numJupiterMoons

materialColor

playerSpeed

// Bad – long, ambiguous


thatFirstThing

theSuperImportantVariableThatMustNotBeNamed

Integral Types Table (C# Reference)

Visual Studio 2015 | [Other Versions](#) ▼


The following table shows the sizes and ranges of the integral types, which constitute a subset of simple types.

Type	Range	Size
sbyte	-128 to 127	Signed 8-bit integer
byte	0 to 255	Unsigned 8-bit integer
char	U+0000 to U+ffff	Unicode 16-bit character
short	-32,768 to 32,767	Signed 16-bit integer
ushort	0 to 65,535	Unsigned 16-bit integer
 int	-2,147,483,648 to 2,147,483,647	Signed 32-bit integer
uint	0 to 4,294,967,295	Unsigned 32-bit integer
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer
ulong	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer

float taxAmount = 0.07f;

VALUE


SUFFIX



Floating-Point Types Table (C# Reference)

Visual Studio 2015 | [Other Versions](#) ▾

The following table shows the precision and approximate ranges for the floating-point types.



Type	Approximate range	Precision
float	$\pm 1.5\text{e}-45$ to $\pm 3.4\text{e}38$	7 digits
double	$\pm 5.0\text{e}-324$ to $\pm 1.7\text{e}308$	15-16 digits

decimal (C# Reference)

Visual Studio 2015 | [Other Versions](#) ▾

The **decimal** keyword indicates a 128-bit data type. Compared to floating-point types, the **decimal** type has more precision and a smaller range, which makes it appropriate for financial and monetary calculations. The approximate range and precision for the **decimal** type are shown in the following table.

Type	Approximate Range	Precision	.NET Framework type
decimal	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / (10^0$ to $28)$	28-29 significant digits	System.Decimal

START

END



```
string quoteOfDay = "Perfect is the enemy of good.";
```