3D Game Programming Geometric Transformations

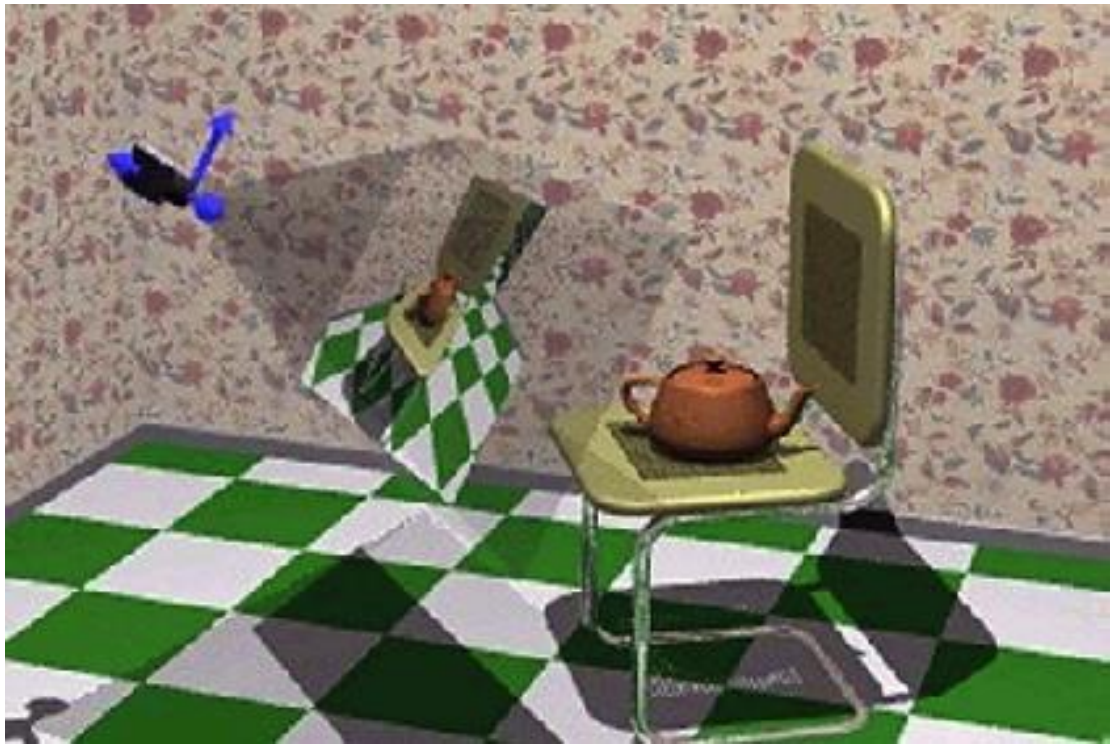
Ming-Te Chi
Department of Computer Science,
National Chengchi University

Outline

- Geometric Transformations
- Basic transformation
  - The coordinates
  - Hierarchy transformation
- Modeling
  - Mesh format / import
  - Unity ProBuilder

Transformation Terminology

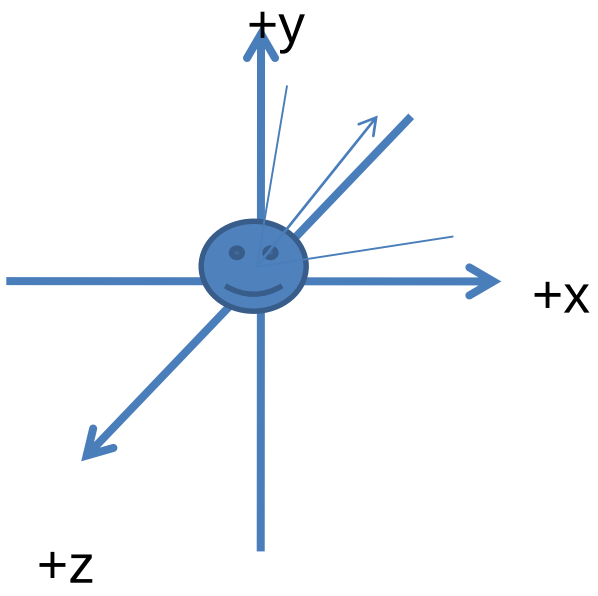
- Viewing
- Modeling
- Modelview
- Projection
- Viewport



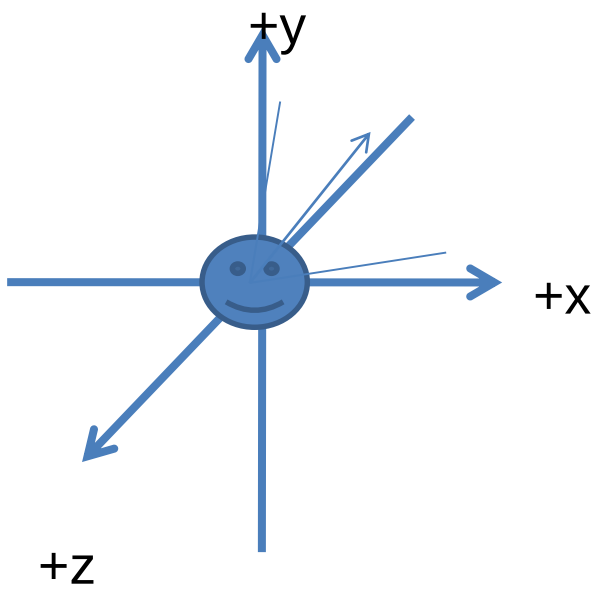
Transformations

- Translation
- Rotation
- Scaling

The Modelview Duality

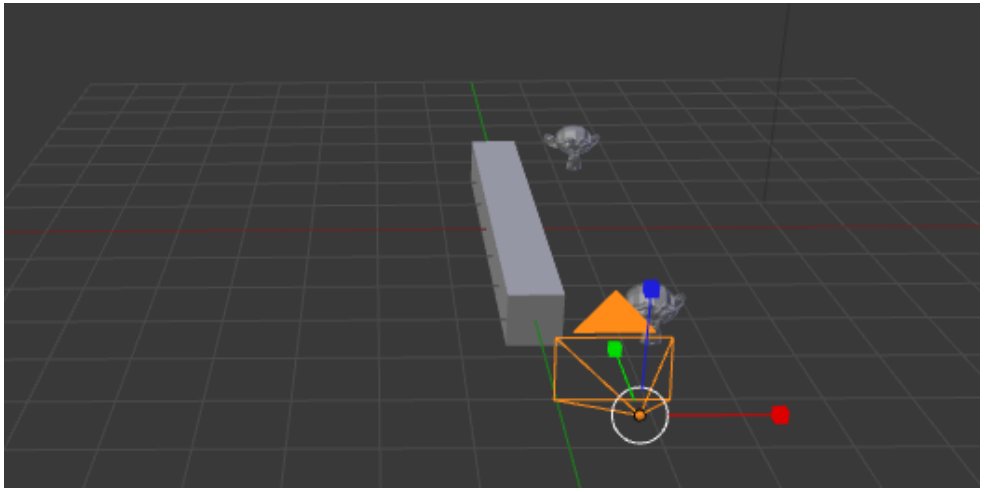


View moving

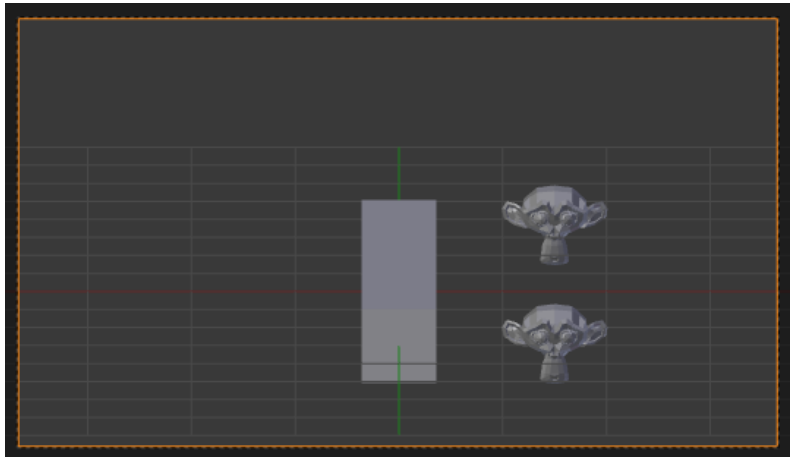


Model moving

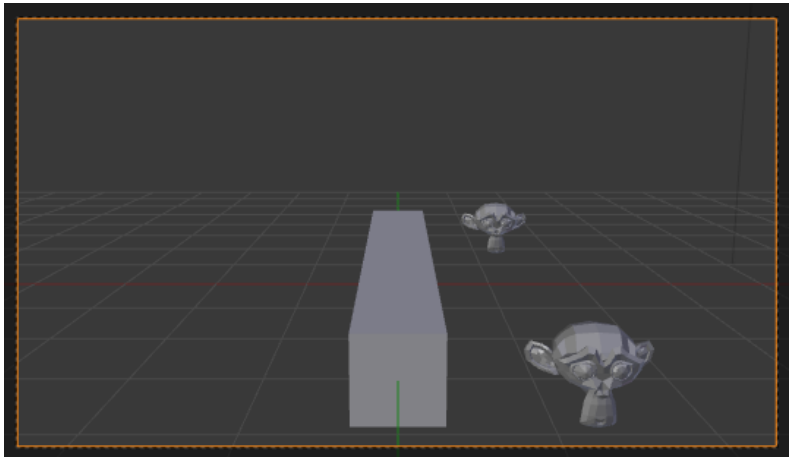
Projection



World space



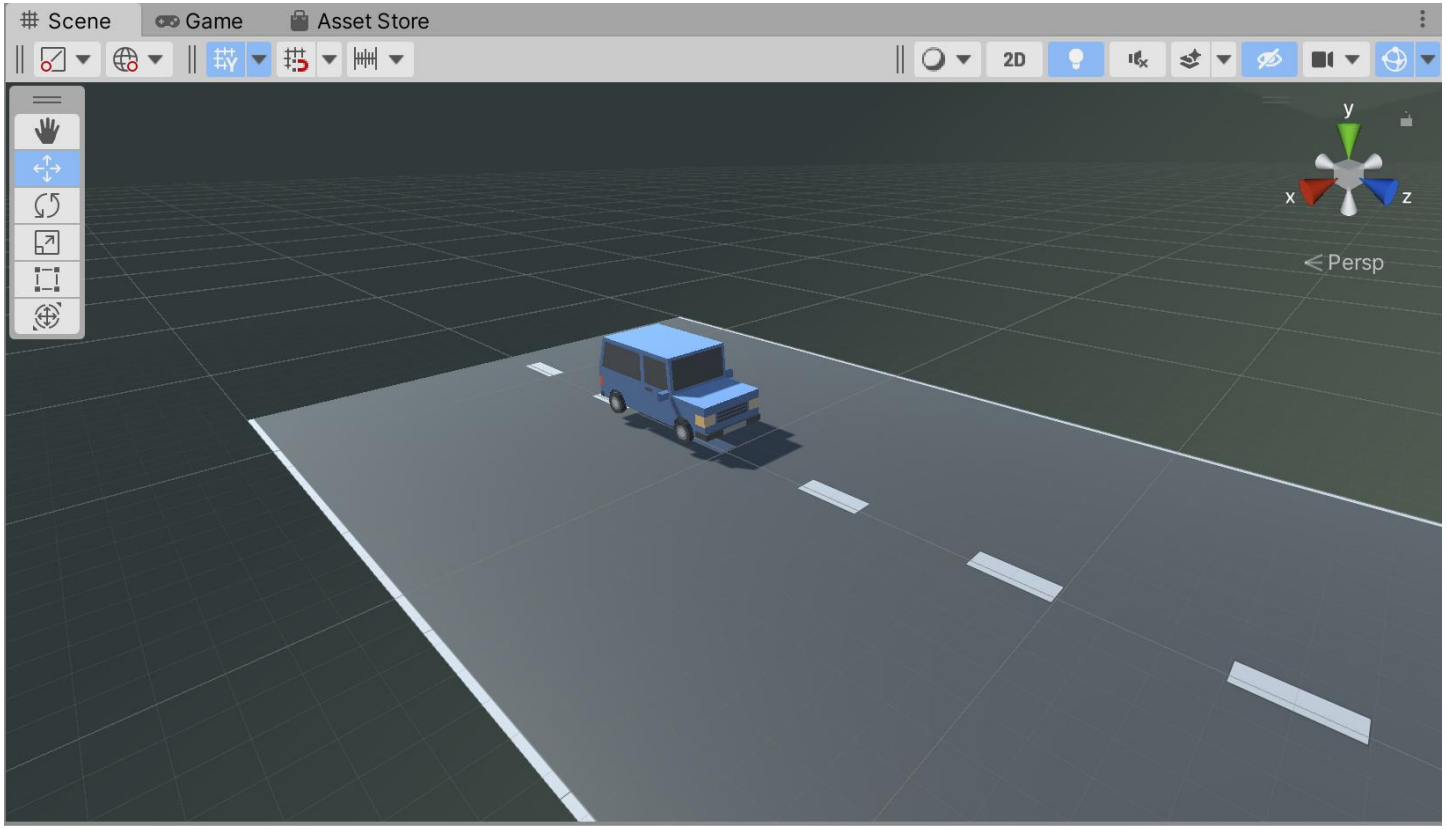
Orthographic



Perspective

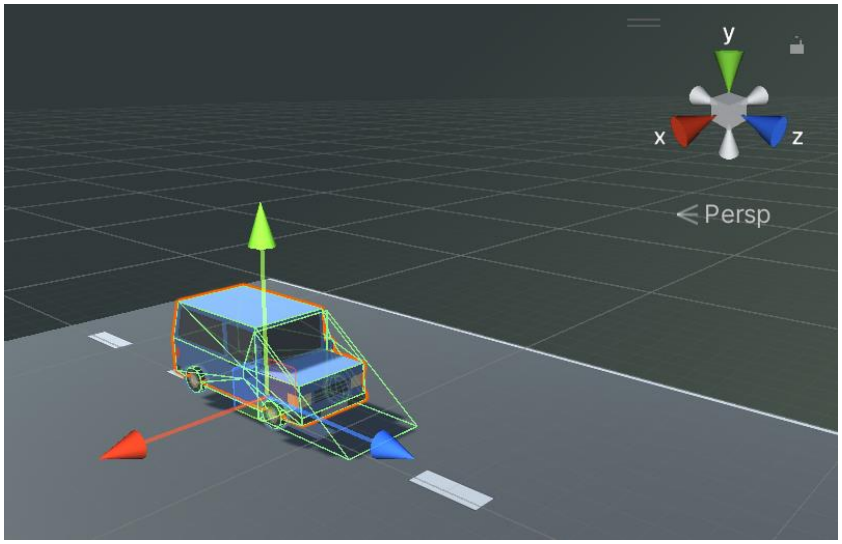
Transform

- Transform used to store and manipulate the position, rotation and scale of the object.



Coordinate System

- Unity is a Left-Handed Coordinate System



right	The red axis of the transform in world space.	X
up	The green axis of the transform in world space	Y
forward	the blue axis of the transform in world space.	Z

Vector3: **Static Properties**

Translate in Unity

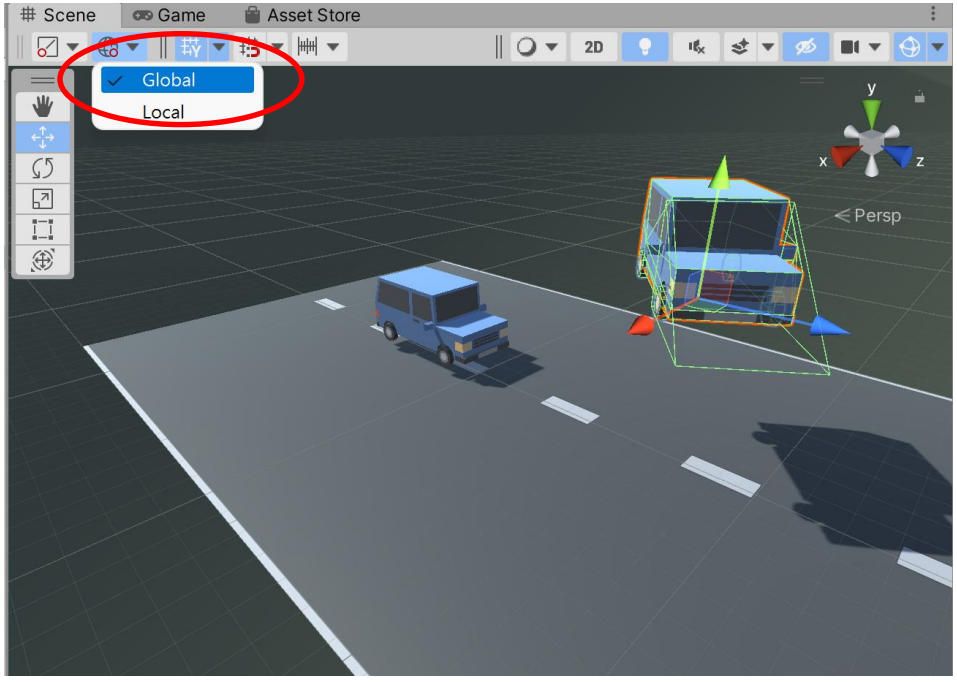
$$\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- public void **Translate**(Vector3 **translation**, Space **relativeTo** = Space.Self);

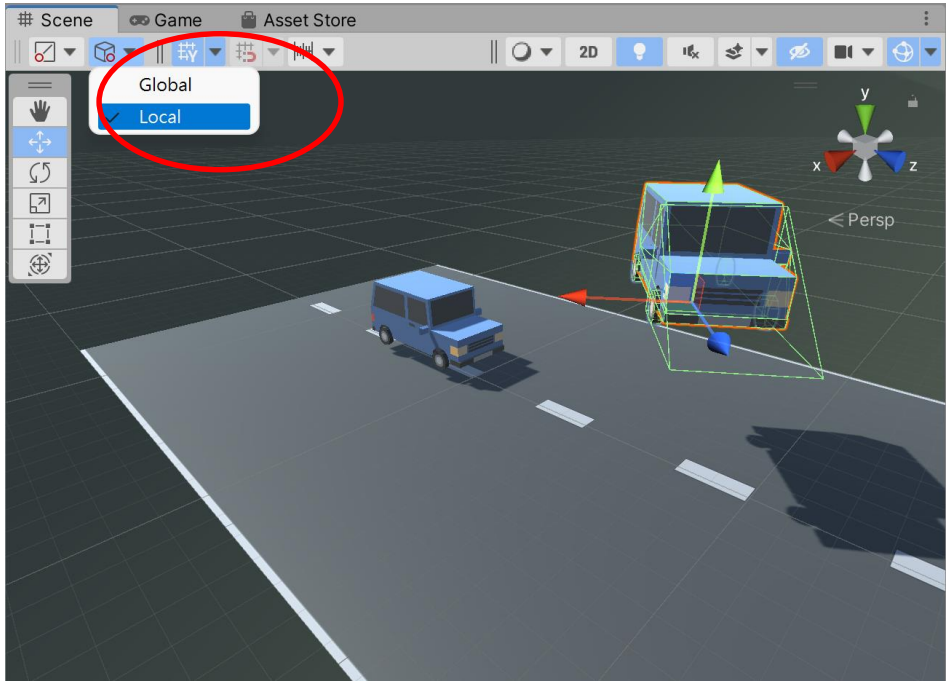
```
// Move the object forward along its z axis 1 unit/second.
transform.Translate(Vector3.forward * Time.deltaTime);

// Move the object upward in world space 1 unit/second.
transform.Translate(Vector3.up * Time.deltaTime, Space.World);
```

- public void **Translate**(float **x**, float **y**, float **z**);

Space



Space.World



Space.Self

Matrix/vector

$$\begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

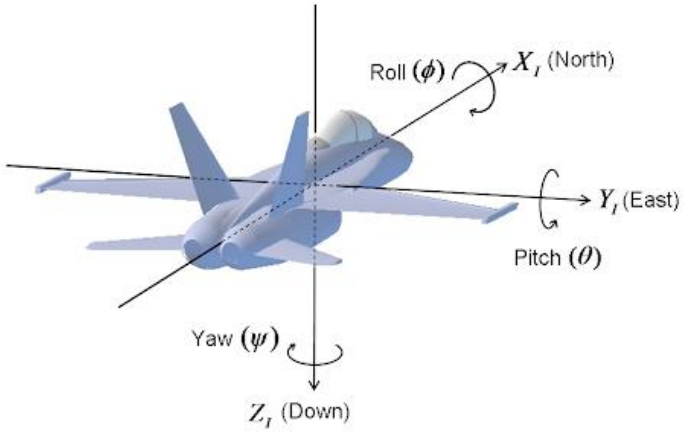
Scale

- public Vector3 localScale;
  - The scale of the transform relative to the GameObjects parent.

```
public float x = 0.1f;
public float y = 0.1f;
public float z = 0.1f;
void Update() {
    // Widen the object by x, y, and z values
    transform.localScale += new Vector3(x, y, z);
}
```

Rotate - Euler Angles

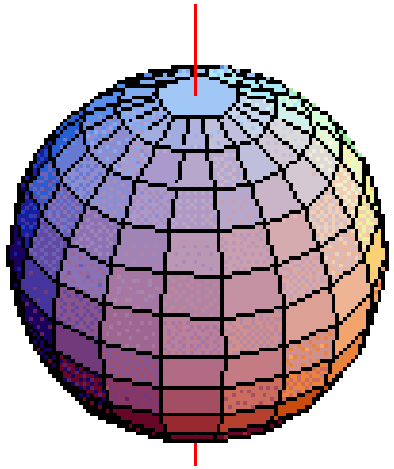
public
void **Rotate**(Vector3 **eulerAngles**, Space **relativeTo** = Space.Self);



Rotate – Axis

public void **Rotate**(Vector3 **axis**,
float **angle**, Space **relativeTo** = Space.Self);

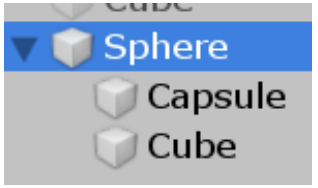
public
void **RotateAround**(Vector3 **point**, Vector3 **axis**,
float **angle**);

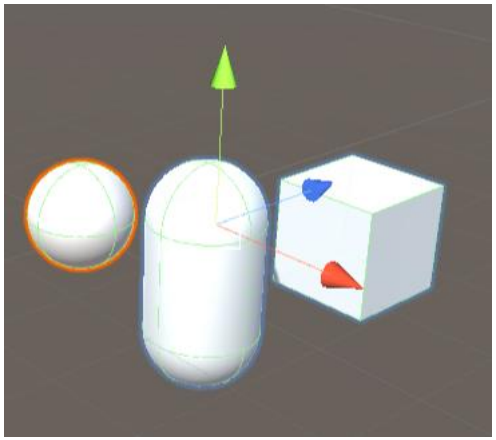


【傅老師】用一顆石頭架場景！！】

Parent/Child

```
// Moves all transform children 10 units upwards!
void Start()
{
    foreach (Transform child in transform)
    {
        child.position += Vector3.up * 10.0f;
    }
}
```





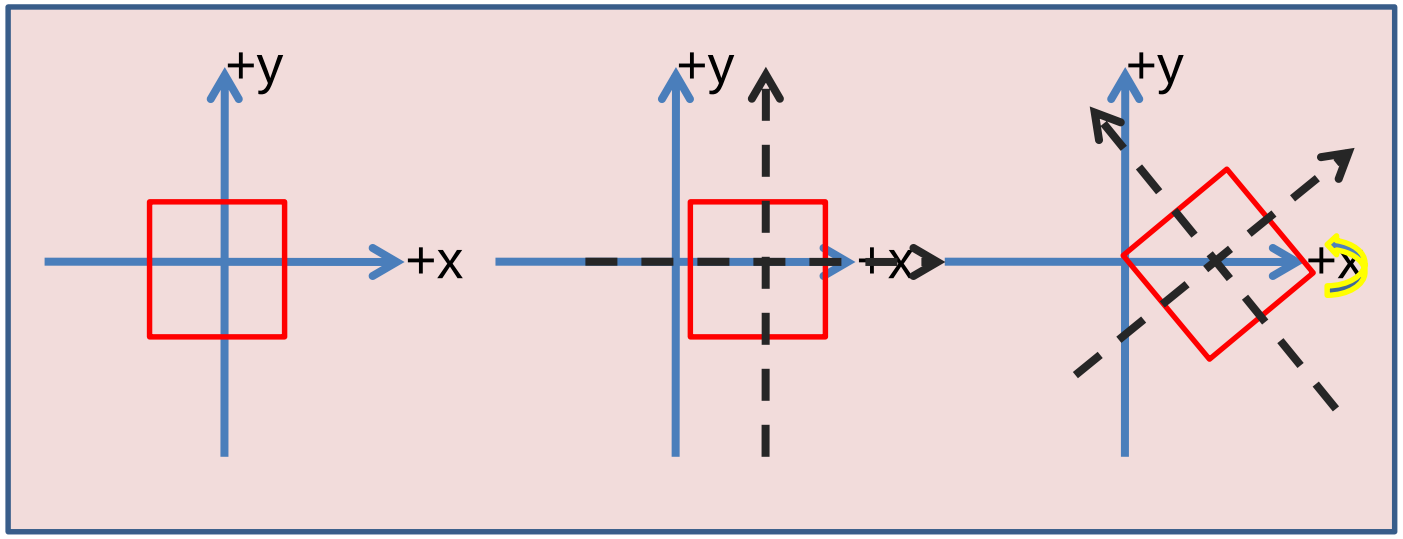
Reset to parent transform

```
transform.rotation = transform.parent.transform.rotation;
transform.localPosition = Vector3.zero;
transform.localScale = Vector3.one;
```

SetParent

```
public void Example(Transform newParent)
{
//Sets "newParent" as the new parent of the player GameObject.
player.transform.SetParent(newParent);

//Same as above, except this makes the player keep its local orientat
//rather than its global orientation.
player.transform.SetParent(newParent, false);
}
```

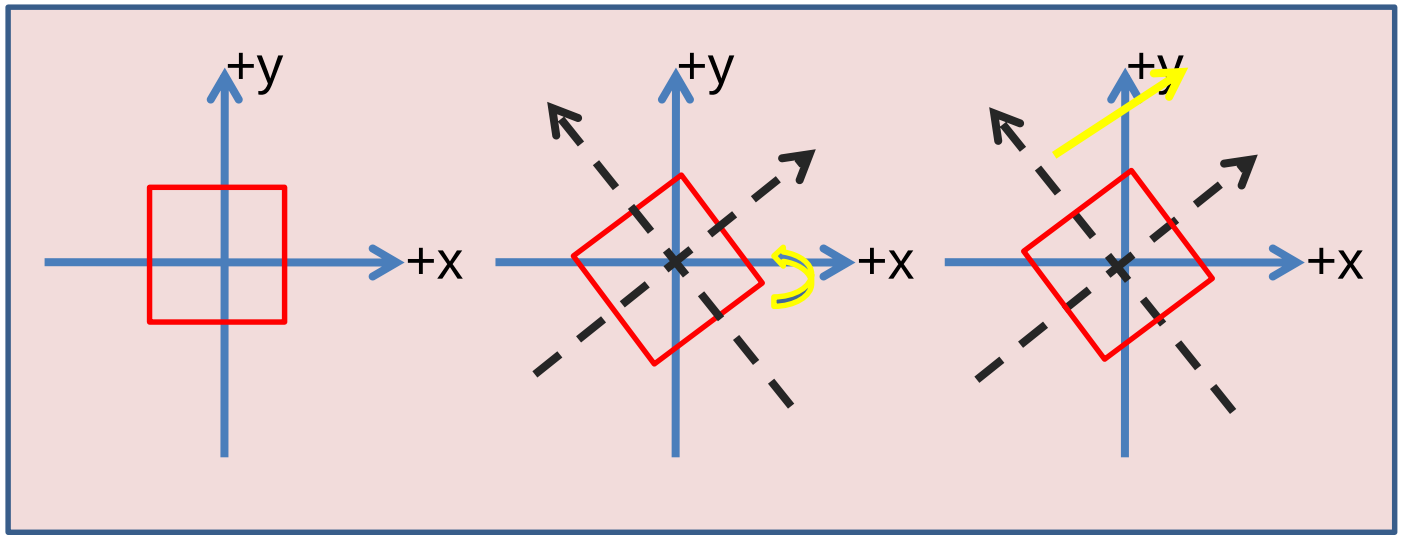
Rotation/Translation
from world to object



Translate()

Rotate()

Rect()

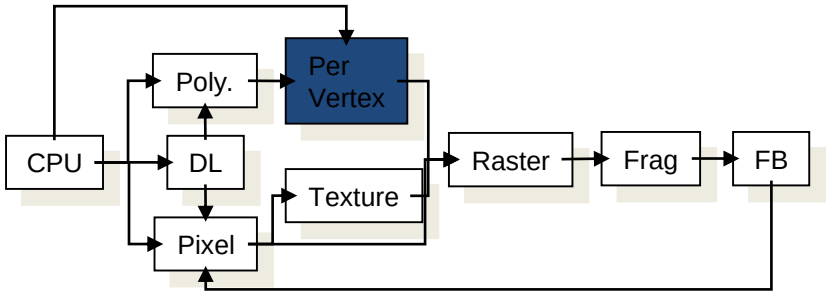


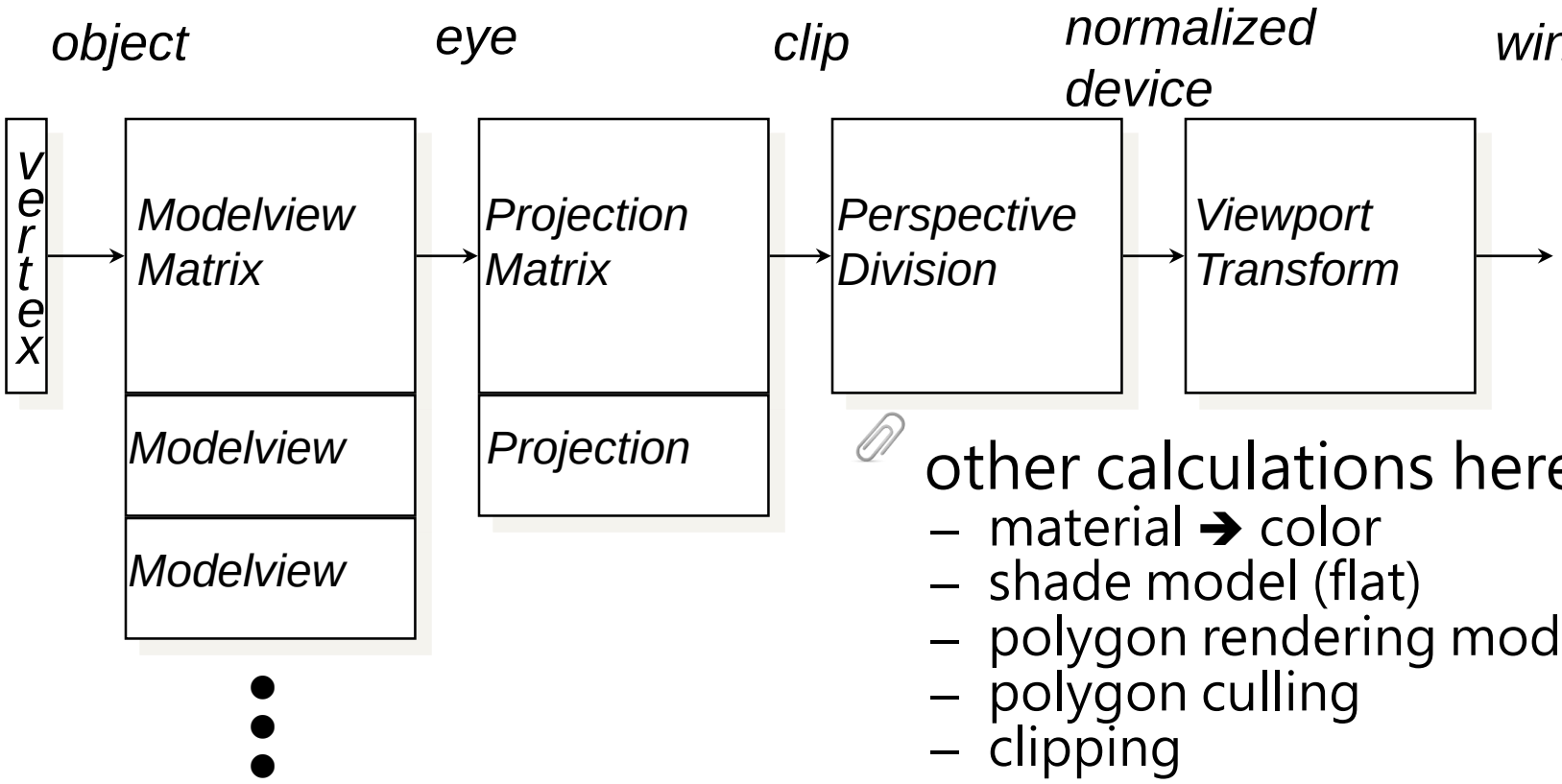
Rotate()

Translate()

Rect()

Transformation Pipeline





other calculations here
- material ➔ color
- shade model (flat)
- polygon rendering mode
- polygon culling
- clipping

The Life of a vertex

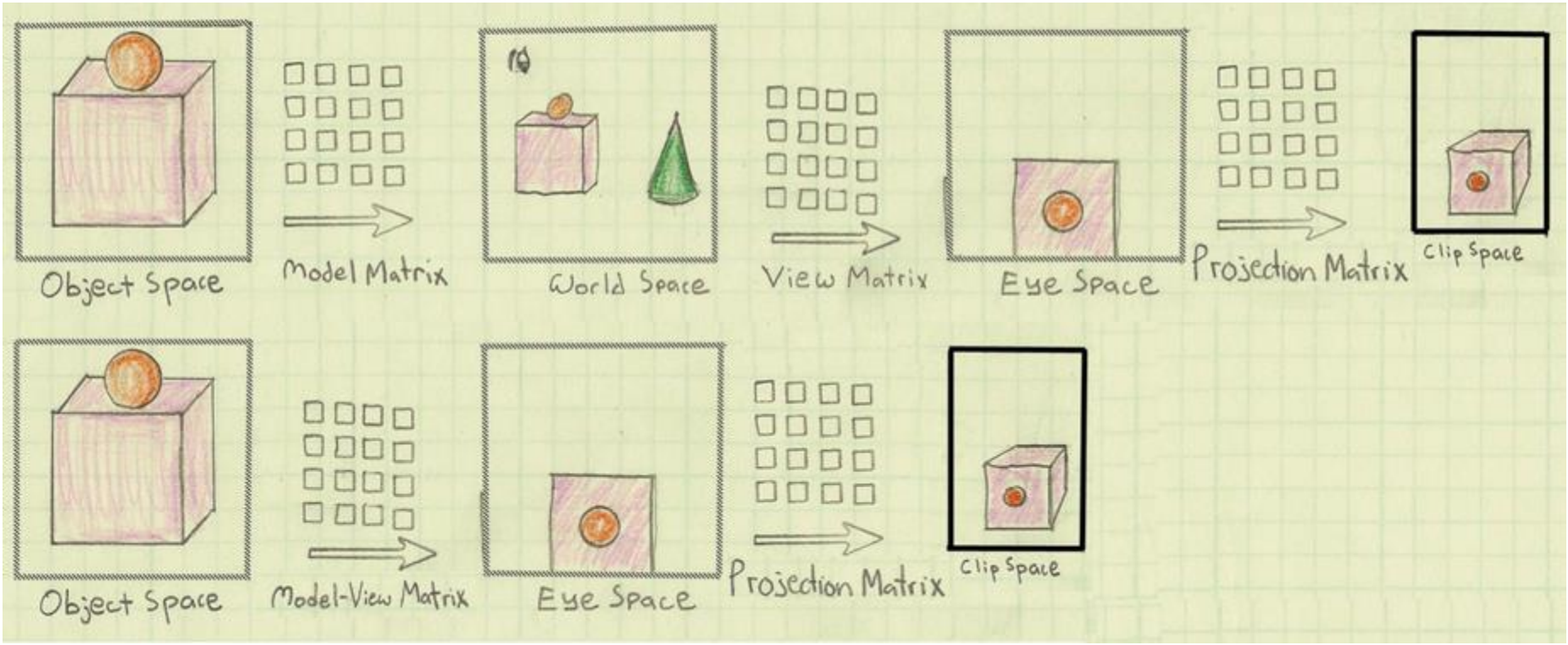
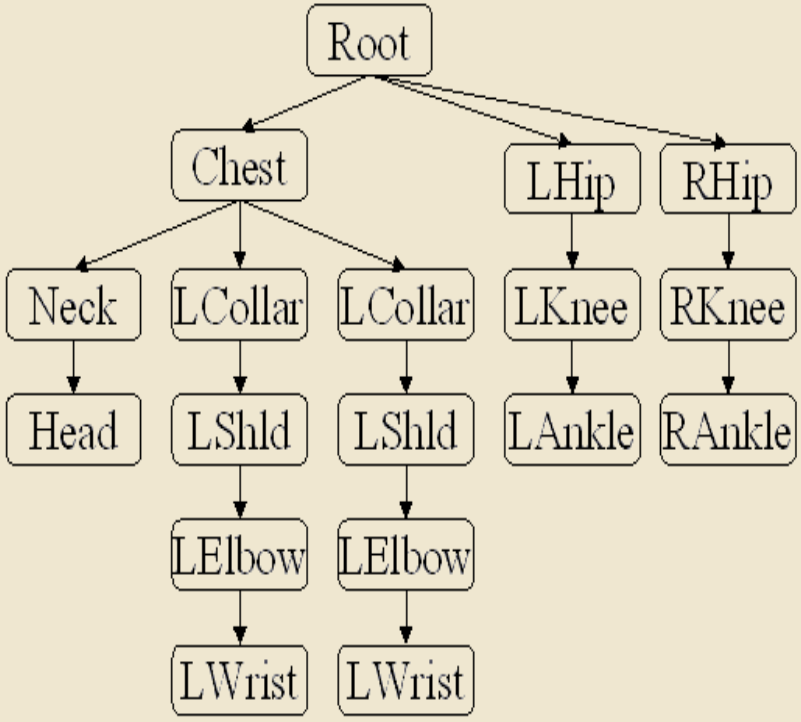
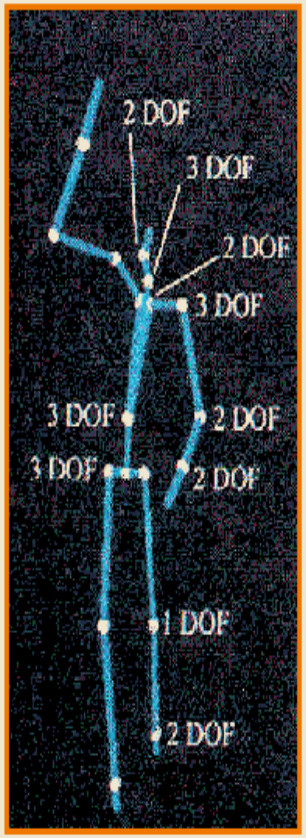


Image by Philp Rideout

Transformation Example 1

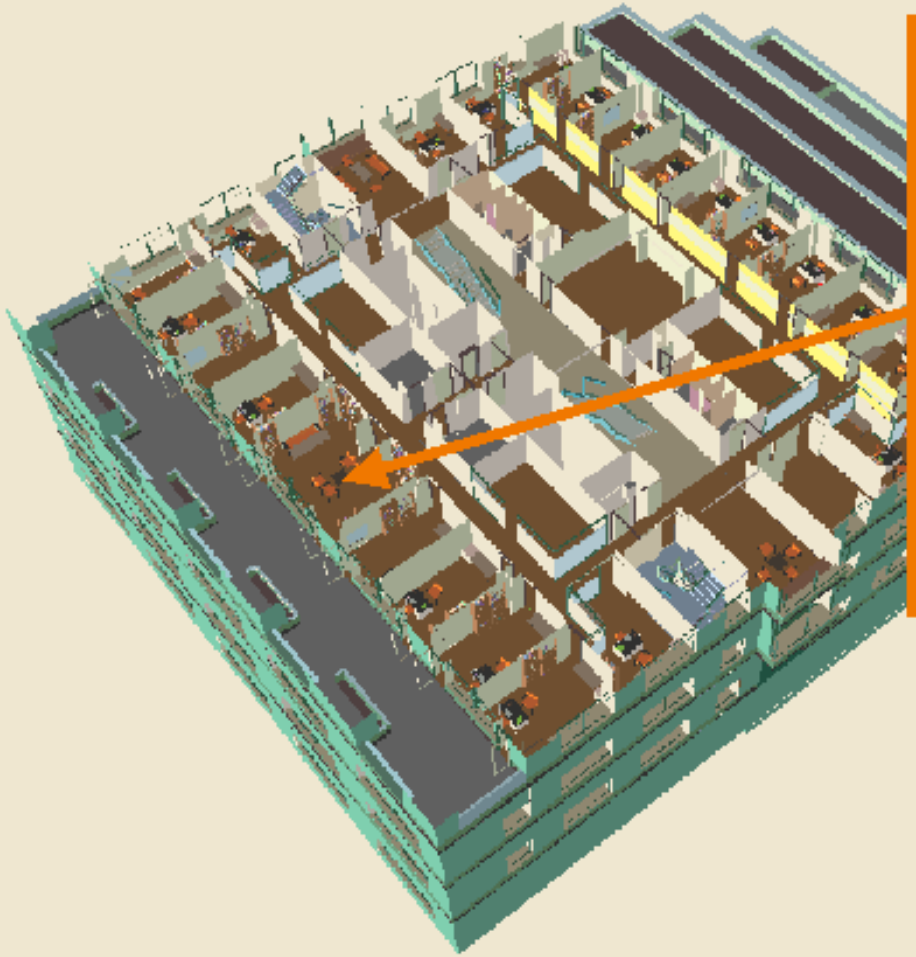
- Well-suited for humanoid characters

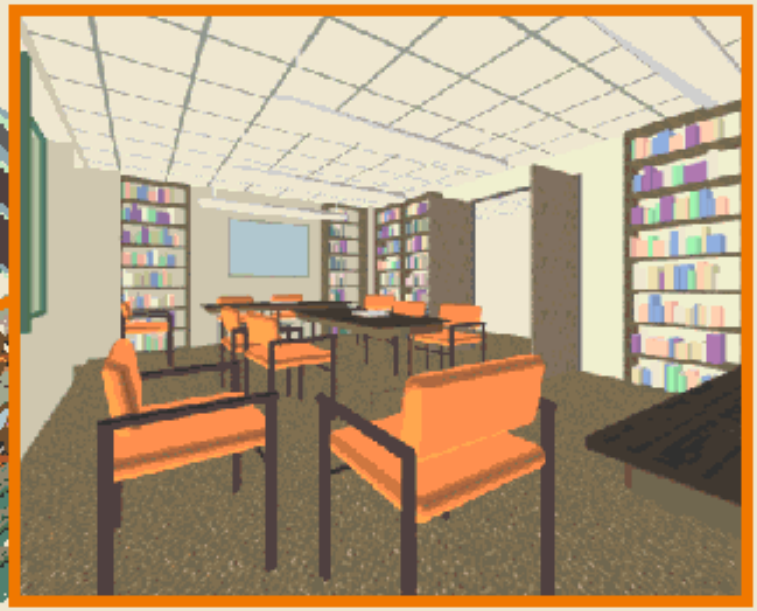




Transformation Example 2

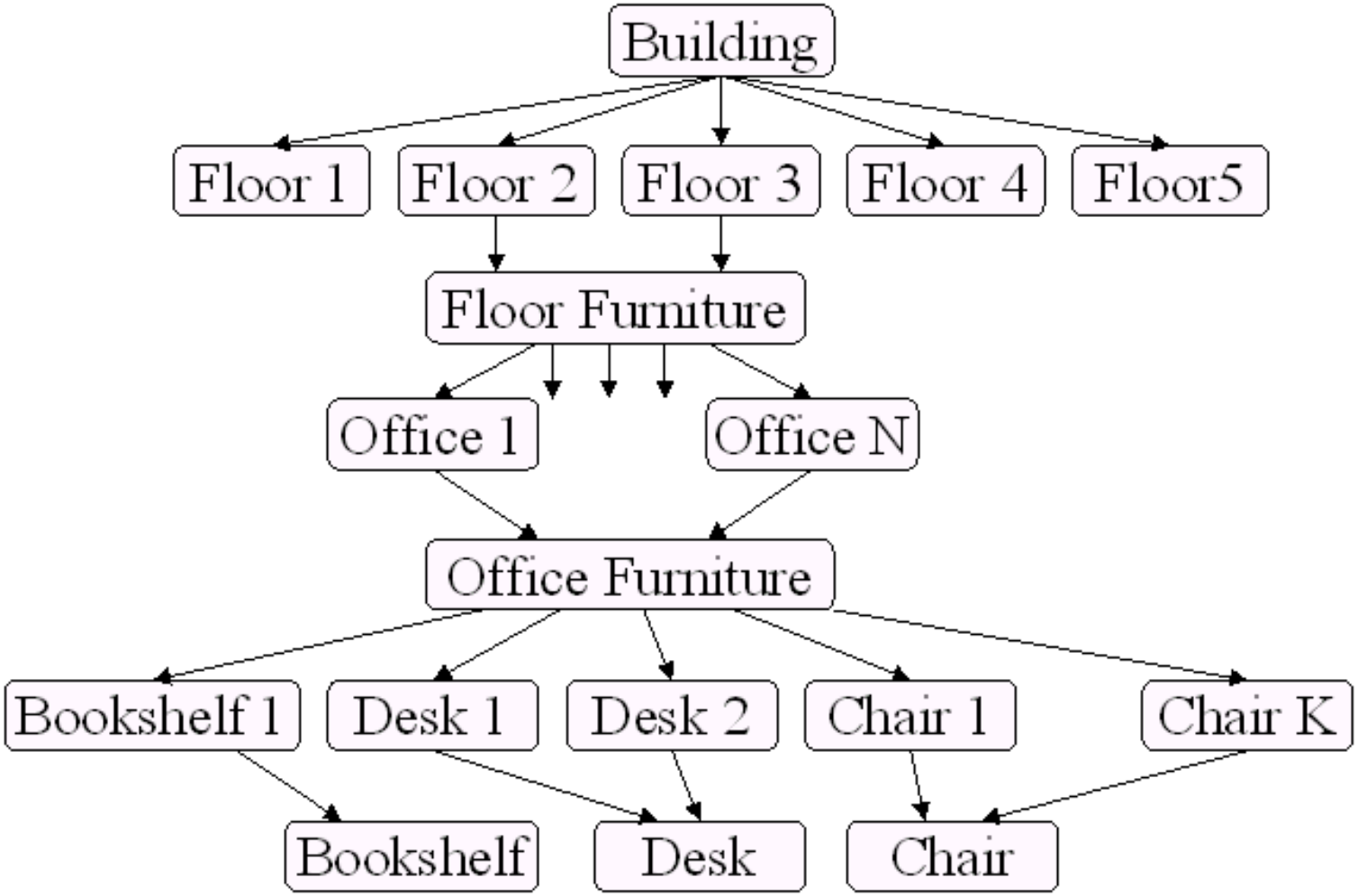
- An object may appear in a scene multiple times





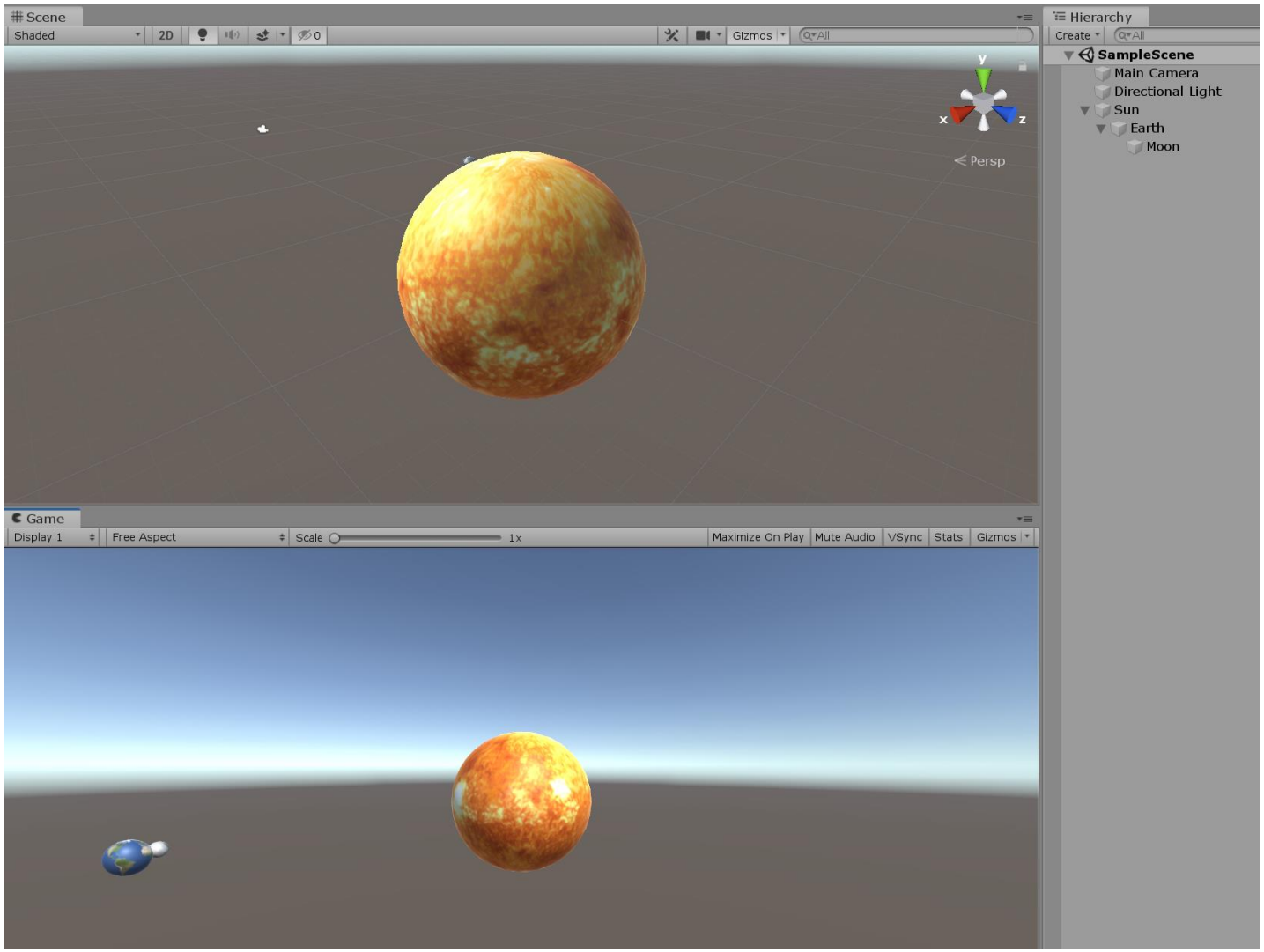
Draw same 3D data with different transformations

Transformation Example 2

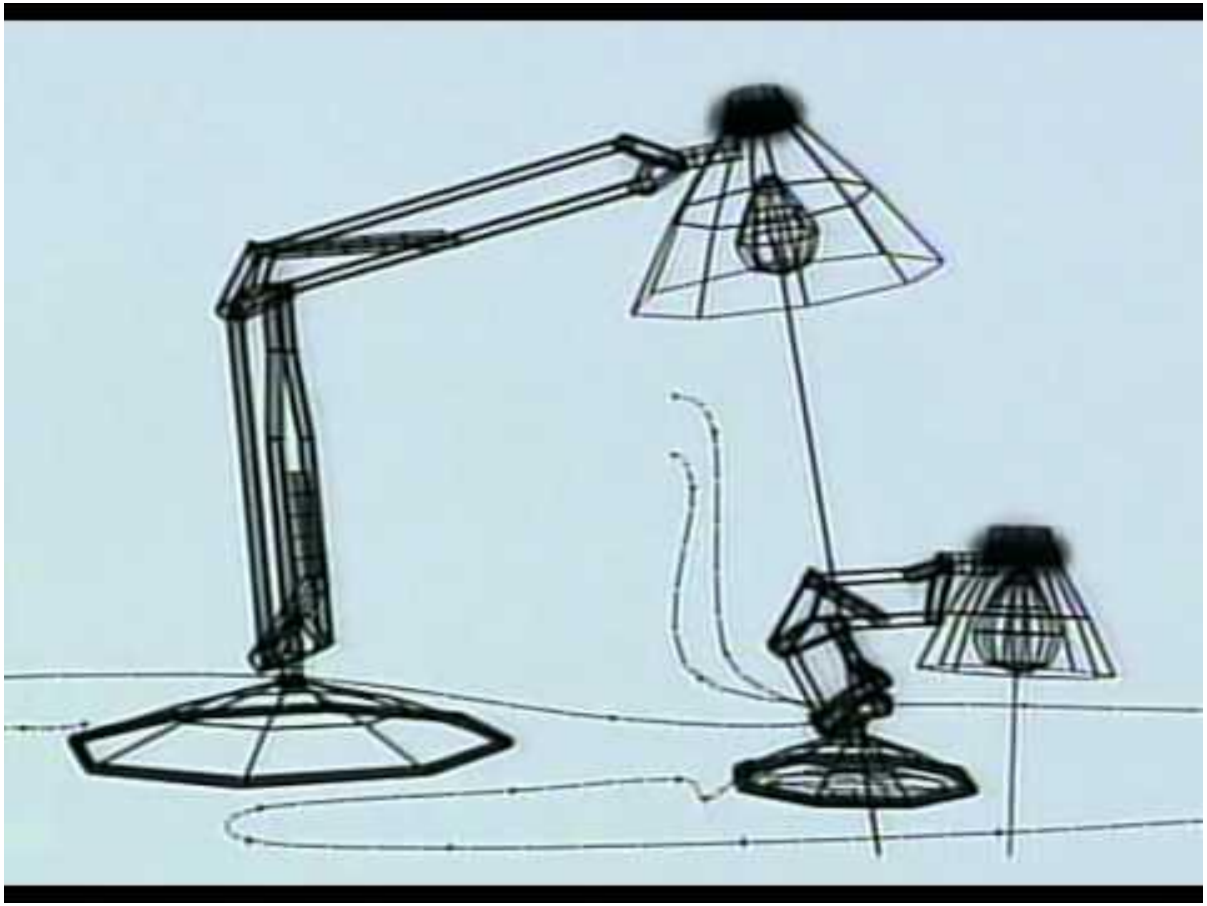


Sun Earth Moon

Assets download



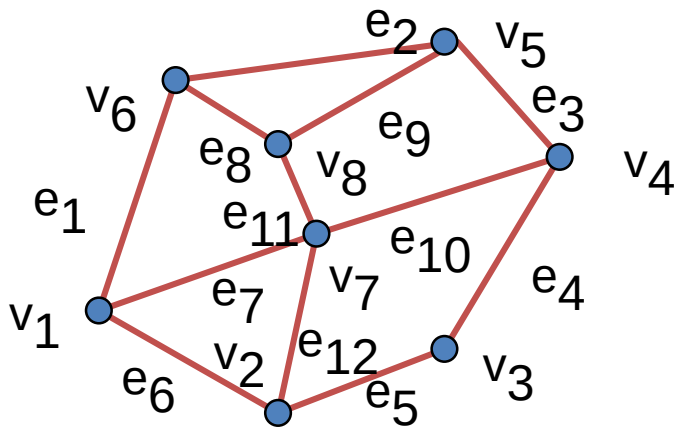
Luxo Jr [Pencil Test] [1986]



MESH FORMAT

Representing a Mesh

- Consider a mesh



- There are 8 nodes and 12 edges
  - 5 interior polygons
  - 6 interior (shared) edges
- Each vertex has a location $v_i = (x_i\ y_i\ z_i)$

3D model format

SIMPLE

- Triangle
  vertex1_X vertex1_Y vertex1_Z normal1_X normal1_Y normal1_Z
  vertex2_X vertex2_Y vertex2_Z normal2_X normal2_Y normal2_Z
  vertex3_X vertex3_Y vertex3_Z normal3_X normal3_Y normal3_Z

COLOR

- Triangle
  frontcolor_R frontcolor_G frontcolor_B backcolor_R backcolor_G backcolor_B
  vertex1_X vertex1_Y vertex1_Z normal1_X normal1_Y normal1_Z
  vertex2_X vertex2_Y vertex2_Z normal2_X normal2_Y normal2_Z
  vertex3_X vertex3_Y vertex3_Z normal3_X normal3_Y normal3_Z

Simple Representation

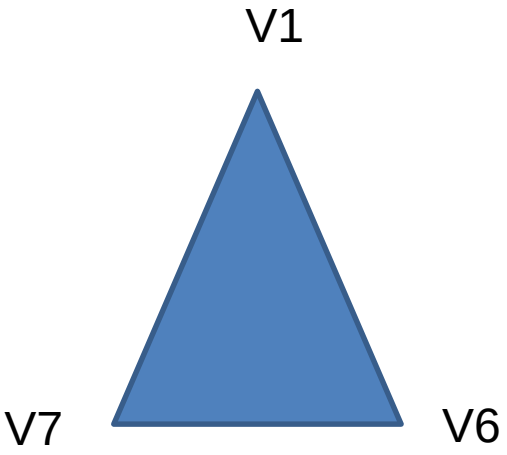
- Define each polygon by the geometric locations of its vertices
- Leads to OpenGL code such as

```
glBegin(GL_POLYGON);
    glVertex3f(x1, x1, x1);
    glVertex3f(x6, x6, x6);
    glVertex3f(x7, x7, x7);
glEnd();
```

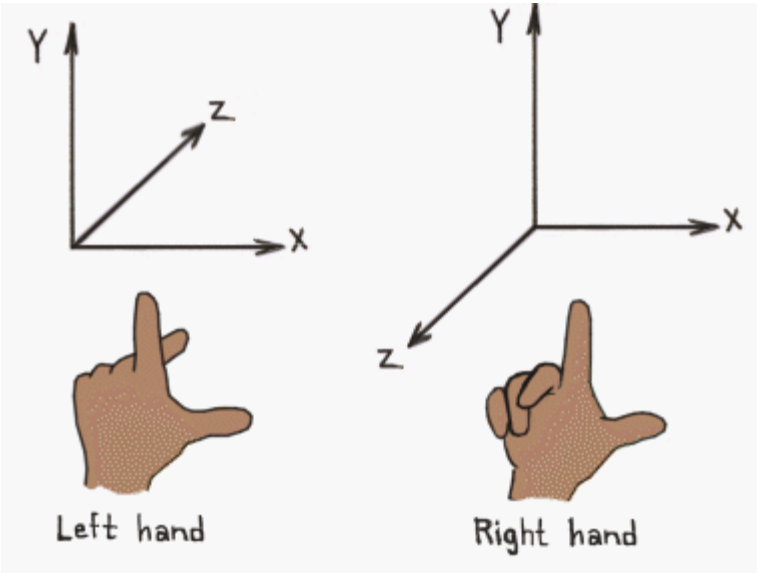
- Inefficient and unstructured
  - Consider moving a vertex to a new location
  - Must search for all occurrences

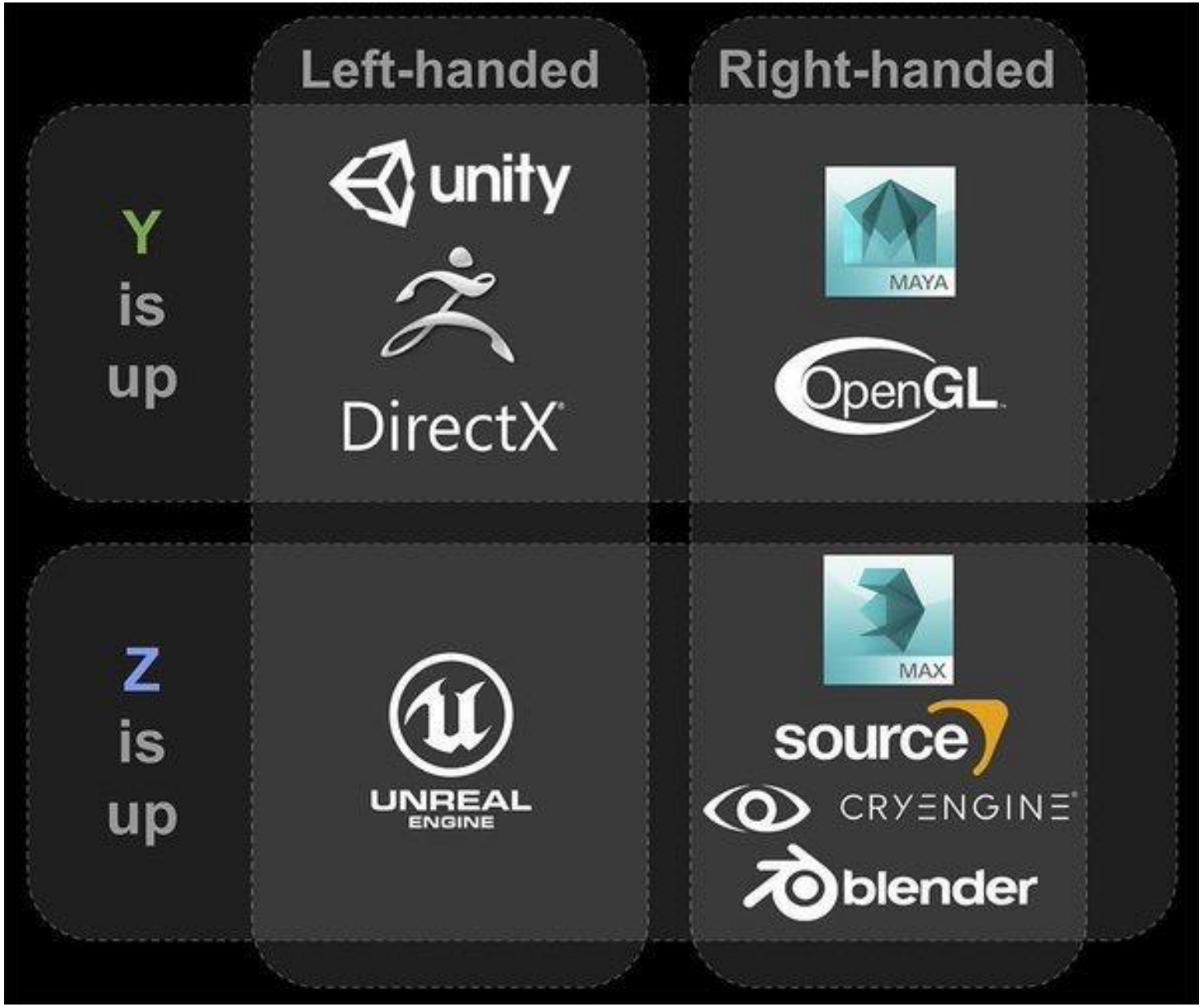
Inward and Outward Facing Polygons

- The order $\{v_1, v_6, v_7\}$ and $\{v_6, v_7, v_1\}$ are equivalent in that the same polygon will be rendered but the order $\{v_1, v_7, v_6\}$ is different.



- Use the *left-hand rule* = clockwise encirclement of outward-pointing normal




Left-handed
Right-handed
Y
is
up
unity
DirectX
MAYA
OpenGL
Z
is
up
UNREAL
ENGINE
MAX
source
CRYENGINE
blender

Wavefront obj format

#example obj file

v -1.63326156 -3.04798102 -8.81131839

....

vn 0.00379090 0.40057179 0.01256634

...

vt 0.22390614 0.97395277 (texture)

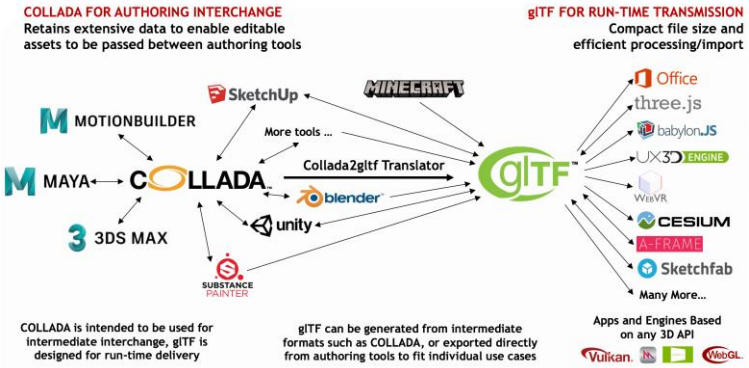
...

f 4/2/4 3/1/3 2/2/2 (index to v/t/n)

Reference

- obj format http://www.martinreddy.net/gfx/3d/OBJ.spec
- ply format
- STL format - http://en.wikipedia.org/wiki/STL_(file_format)

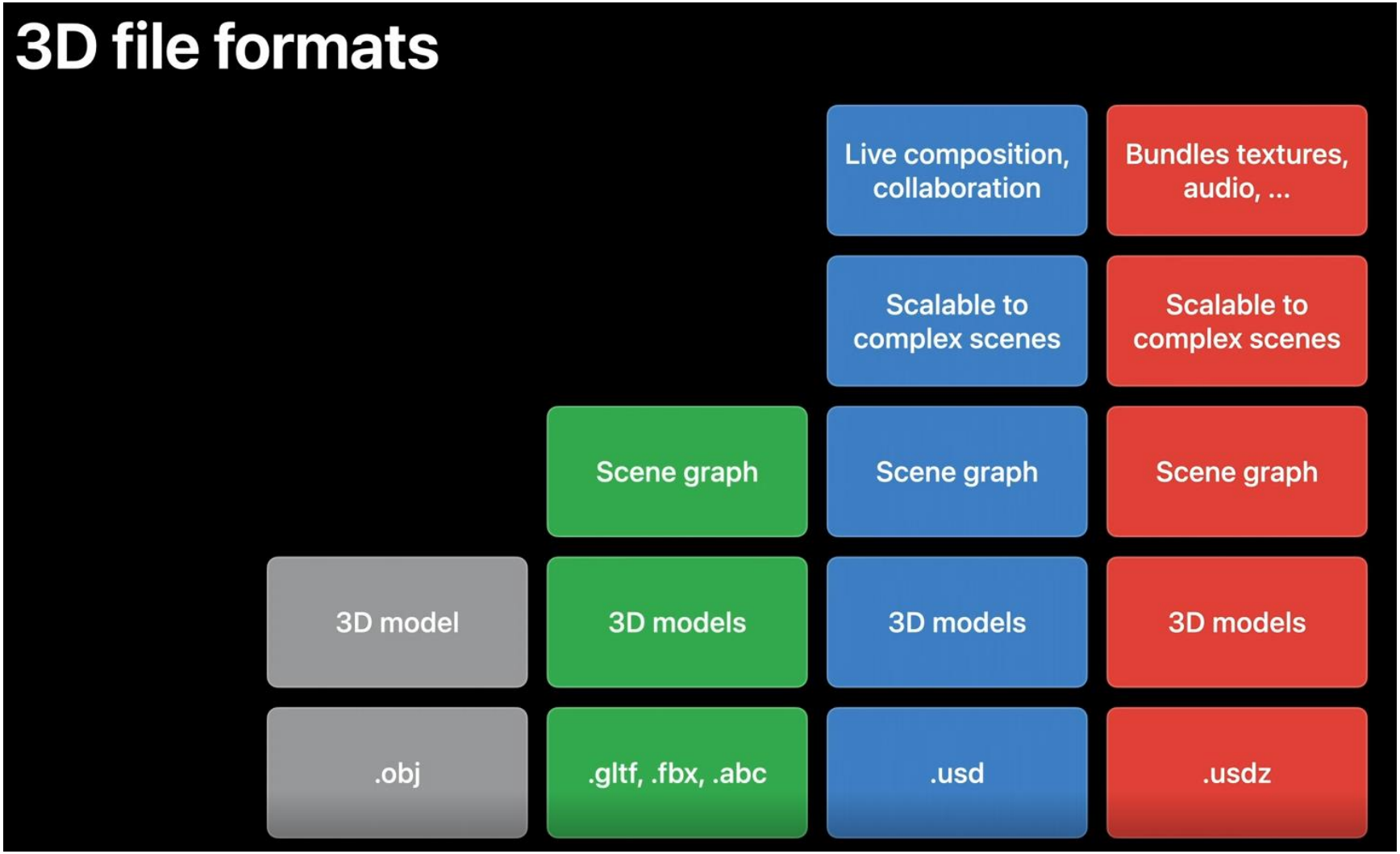
Scene Graph



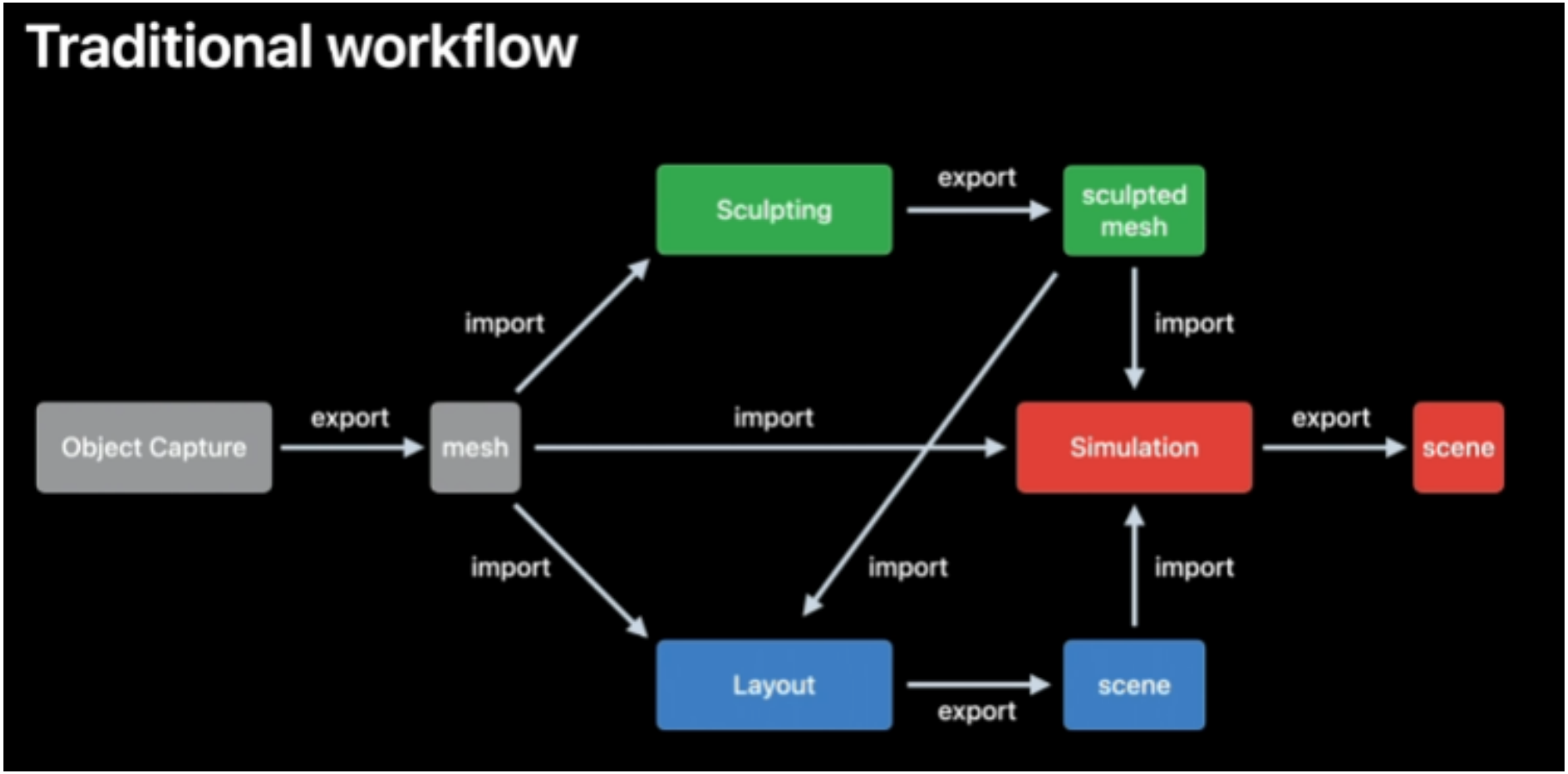
- FBX (**Film**box) by Autodesk
- COLLADA (***COLLA**borative **D**esign **A**ctivity*)
- Alembic by sony

- USD (Universal Scene Description) by pixar
  - Five Things to Know About USD

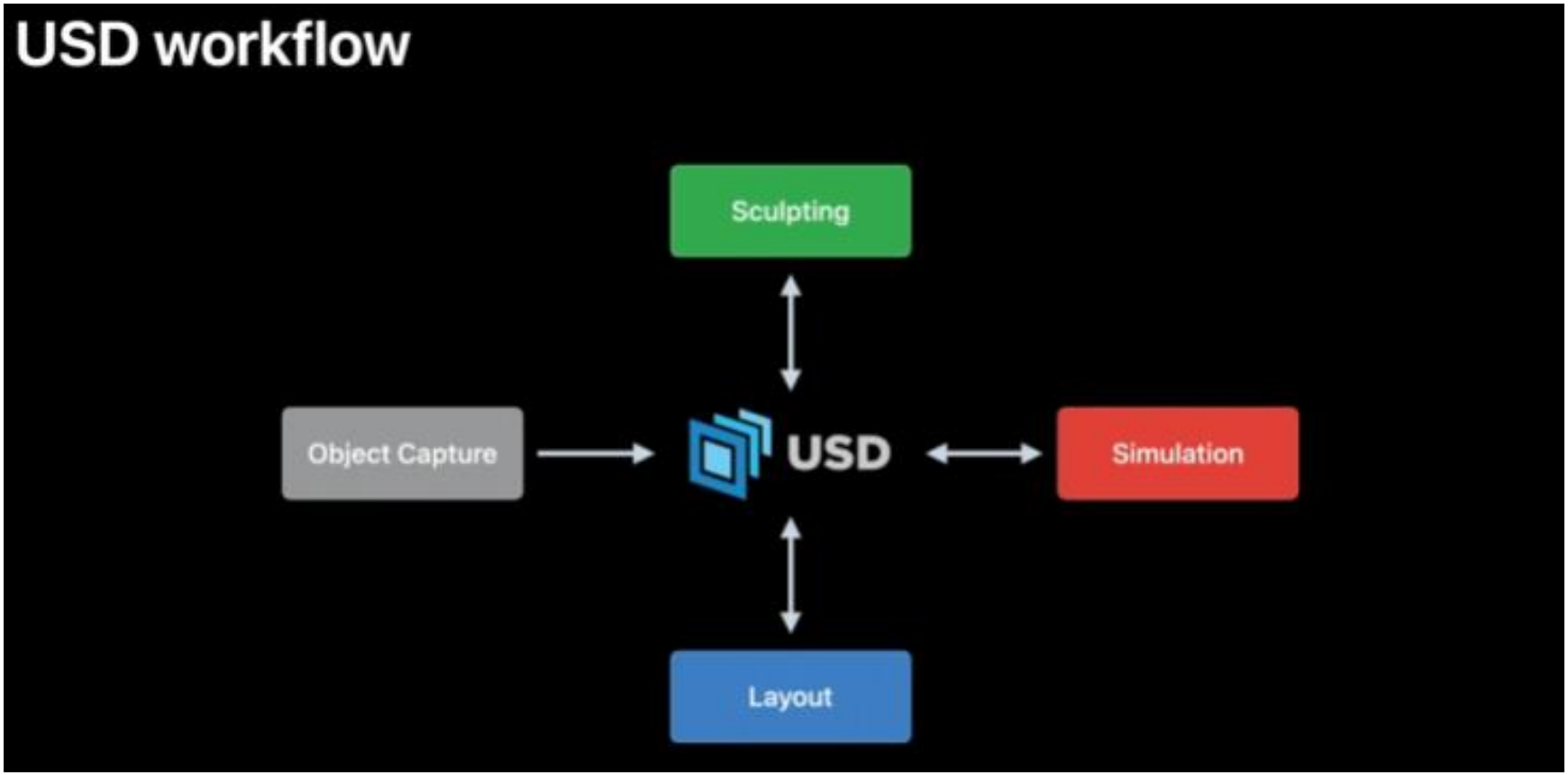
Universal Scene Description, USD (WWDC 2021)



Create 3D workflows with USD (WWDC 2021)



Create 3D workflows with USD (WWDC 2021)



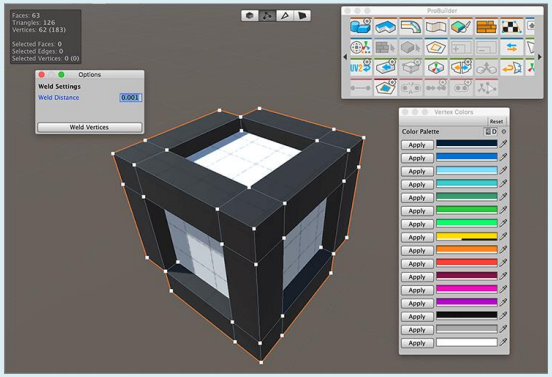
Create 3D workflows with USD (WWDC 2021)

PROBUILDER

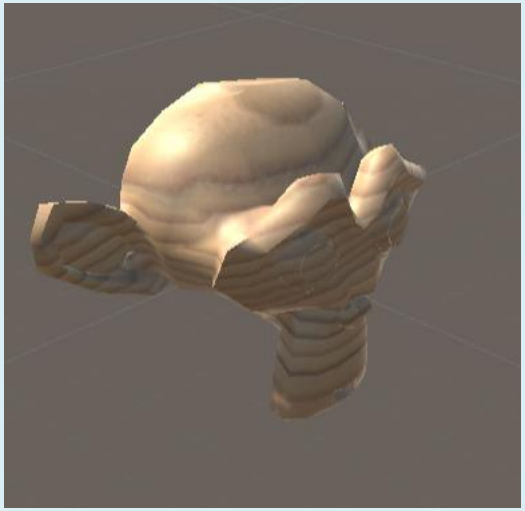
Building and editing complex Meshes inside Unity

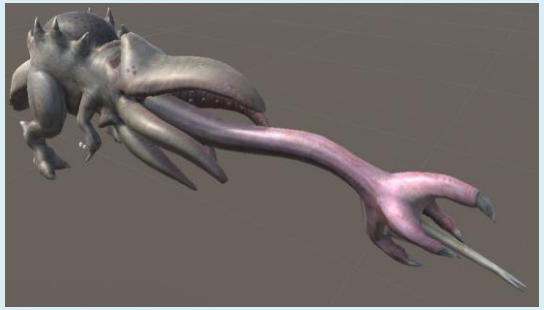
Modeling

Assemble
with Primitive



ProBuilder
inside Unity

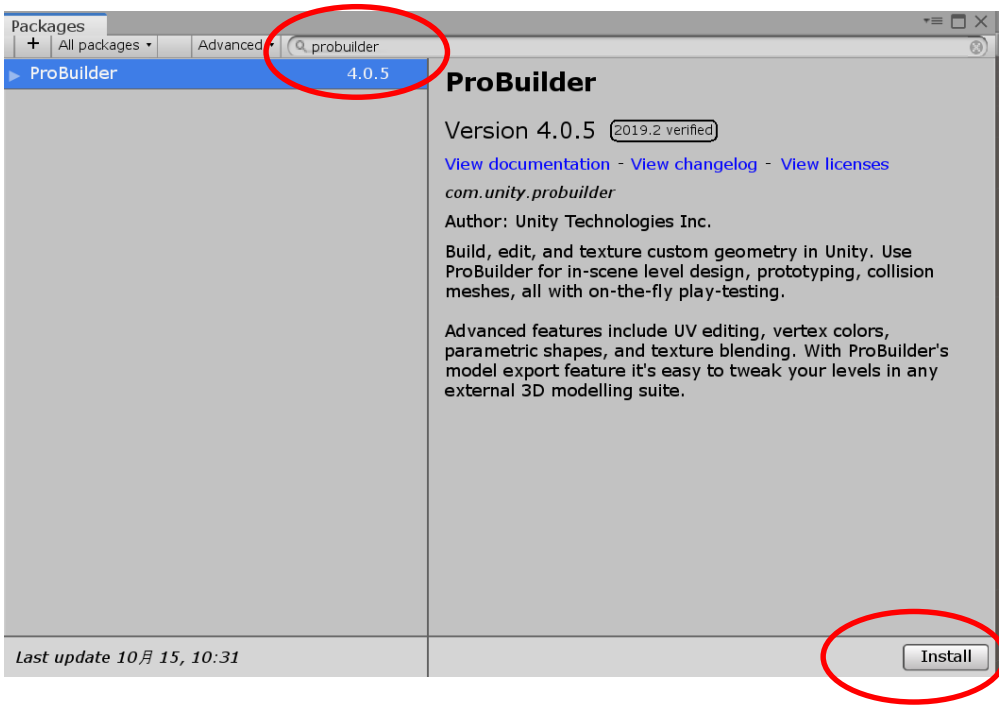




Import
from modeling
tools

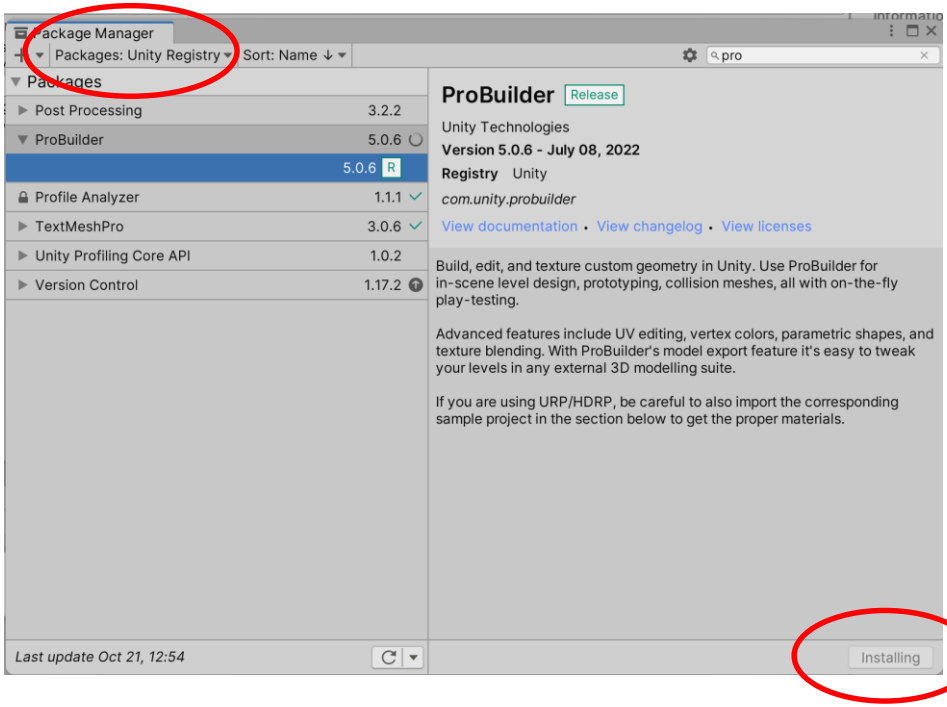
Installing ProBuilder

- Open the Package Manager (in Unity's top menu: **Window** > **Package Manager**).
- Enter **ProBuilder** in the search box.
- Click **ProBuilder** in the package list (left side), then click the **Install** button in the package details (right side).



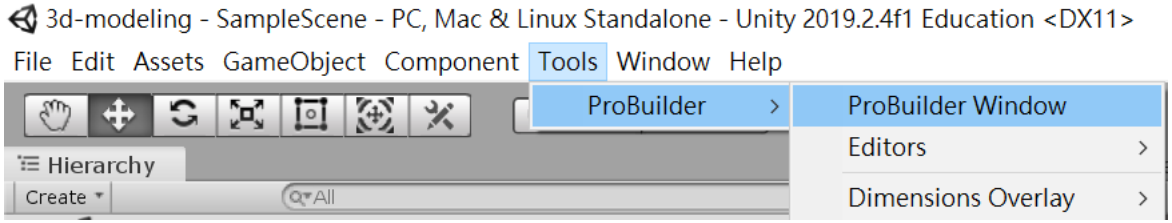
Installing ProBuilder

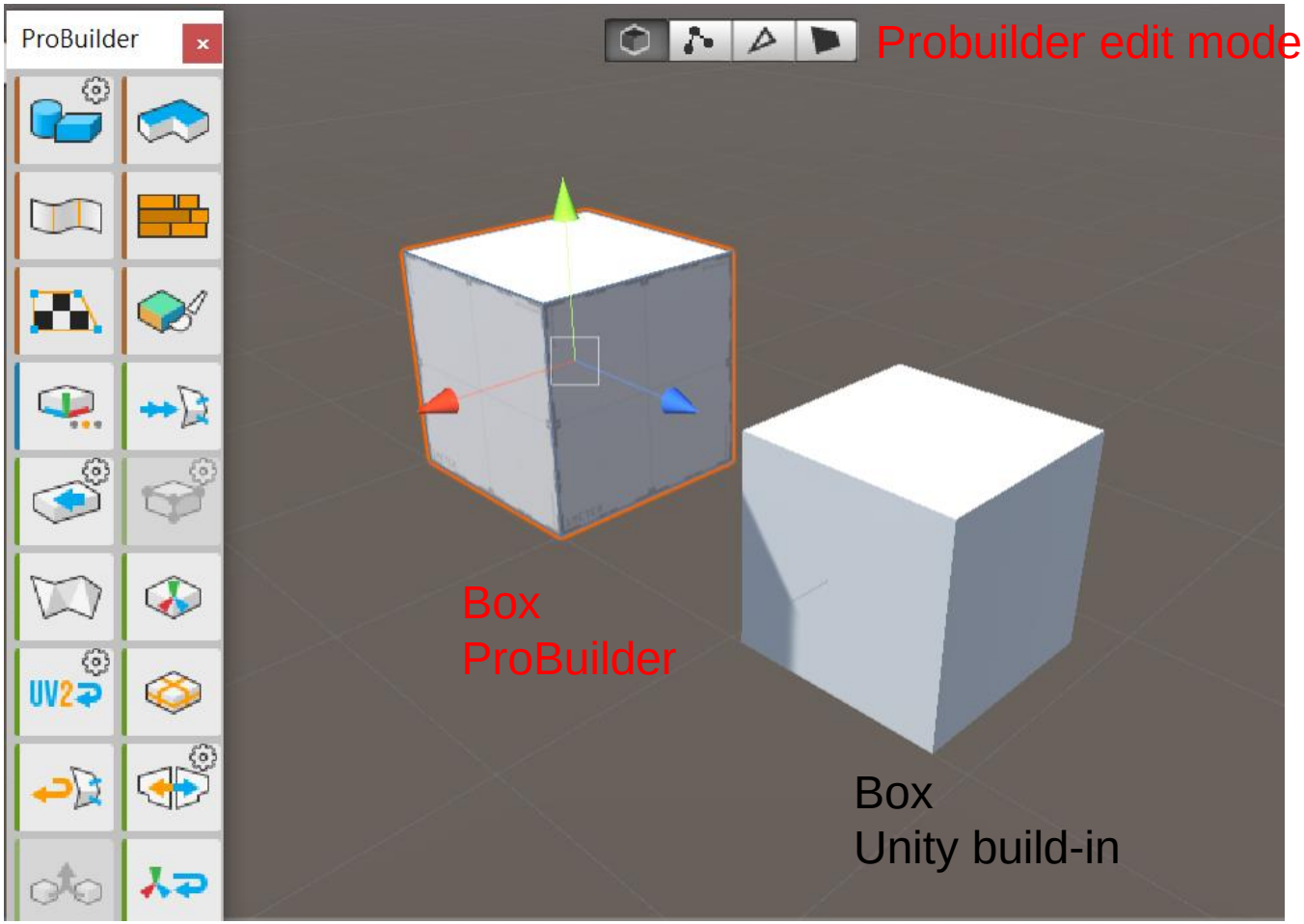
- Open the Package Manager (in Unity's top menu: **Window** > **Package Manager**).
- Enter **ProBuilder** in the search box.
- Click **ProBuilder** in the package list (left side), then click the **Install** button in the package details (right side).



To verify that ProBuilder is correctly installed, open the ProBuilder toolbar (from Unity's top menu: **Tools** > **ProBuilder** > **ProBuilder Window**).

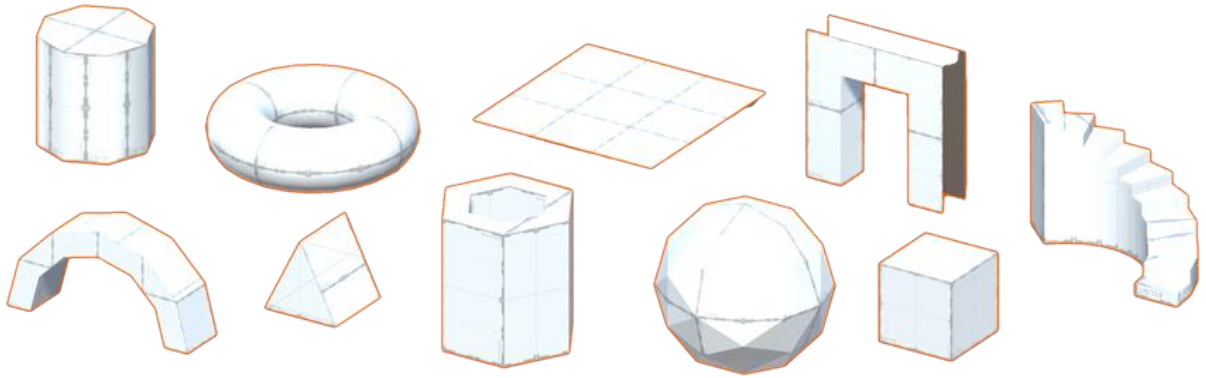
If you don't see the ProBuilder Window menu item, then ProBuilder did not install correctly.

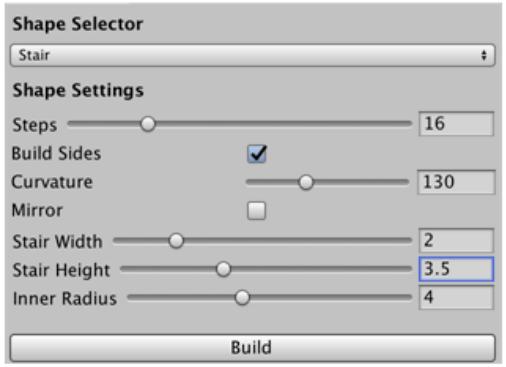


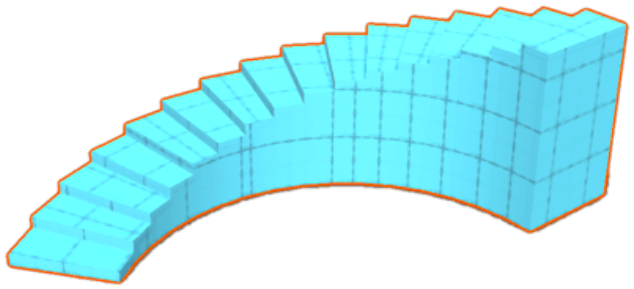


Probuilder toolbar

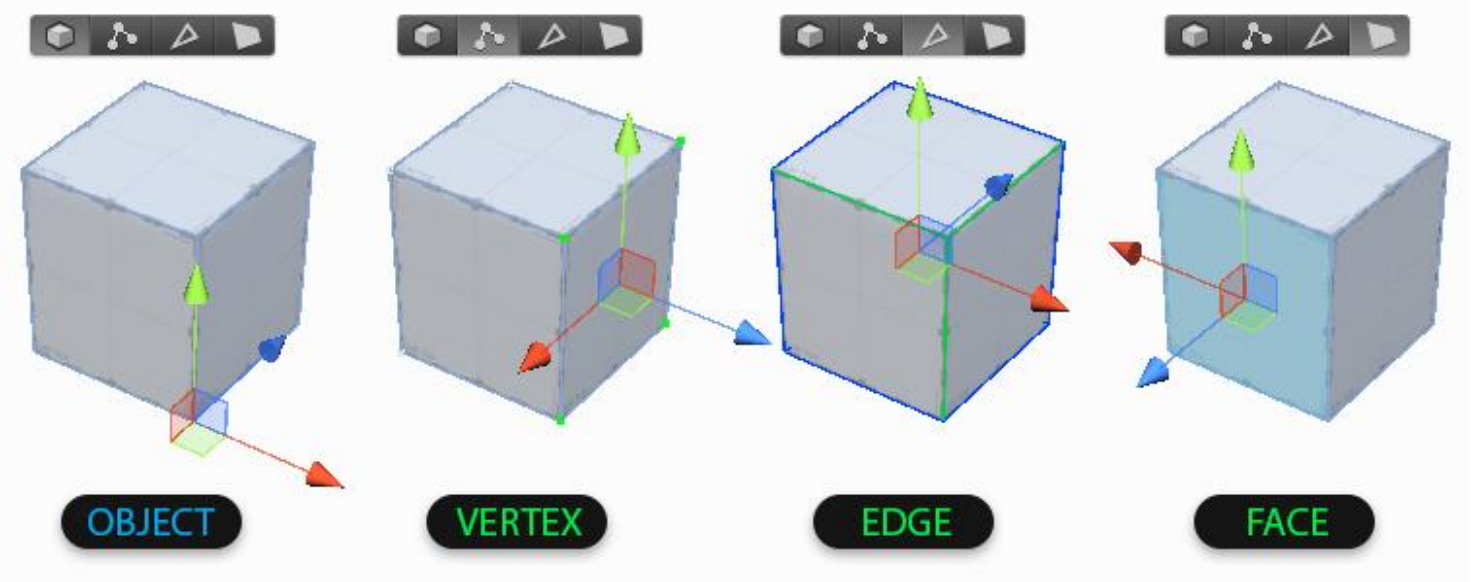
Creating Meshes



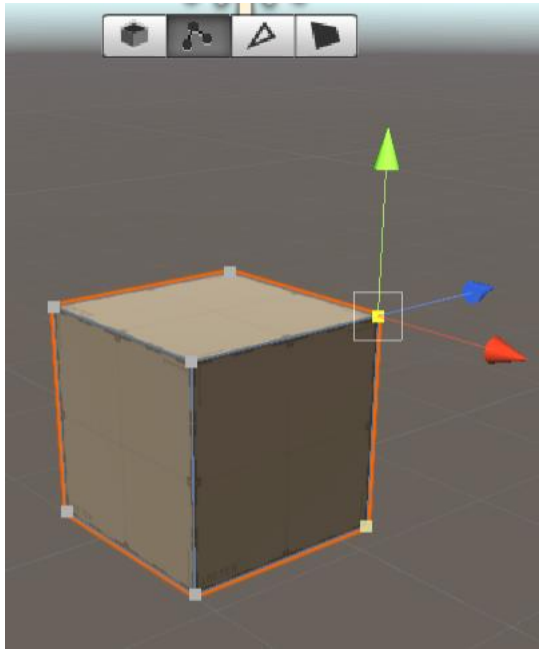


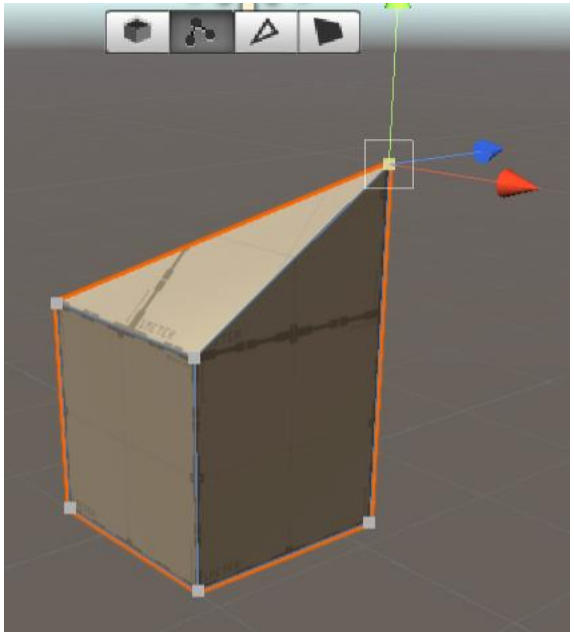


Edit modes (Object vs Element)

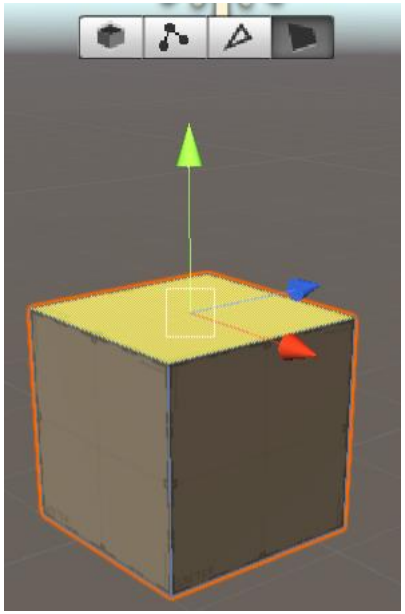


Select -> Transforming



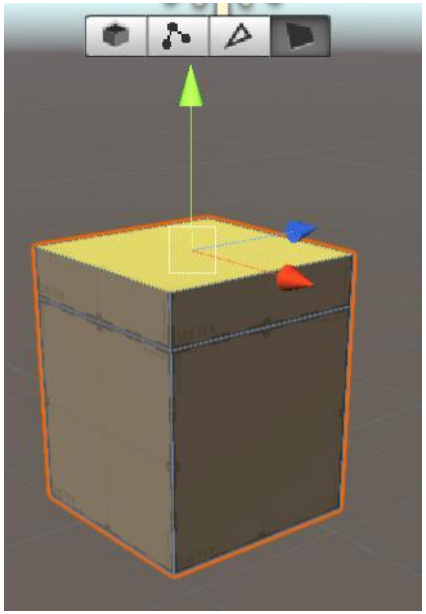


Select face -> Extruding



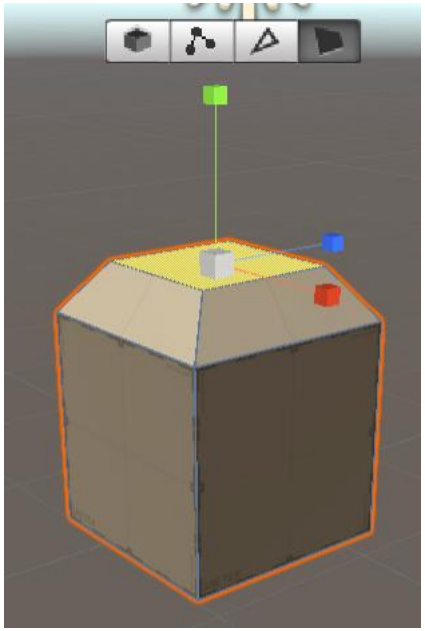
Select a face

Extruding



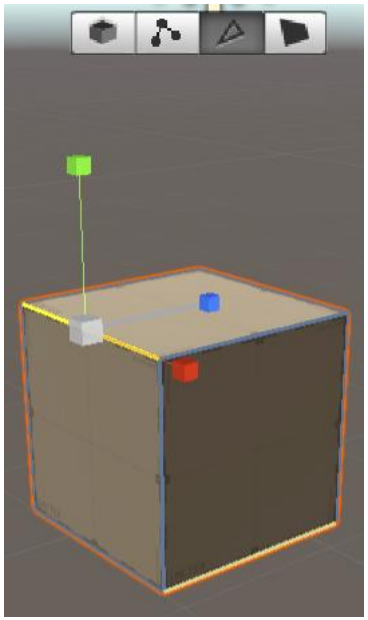
Shift key +
Translate

Insetting

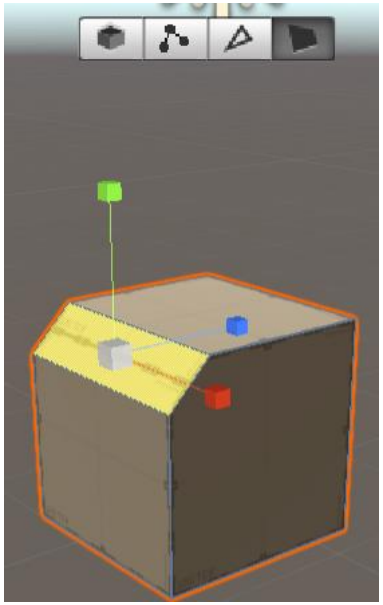


Shift key +
Scale

Select edge -> Bevel

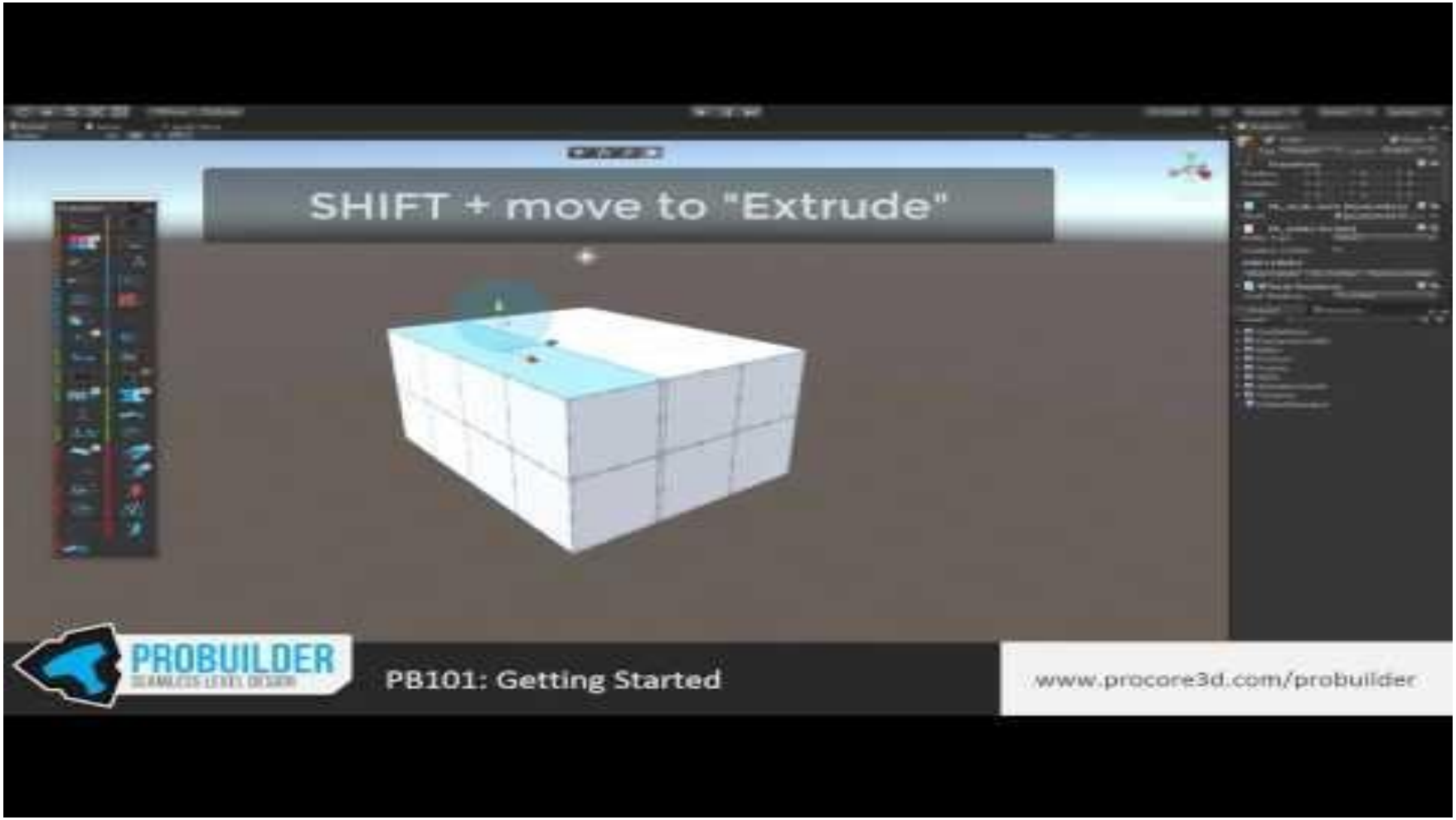


Select a edge

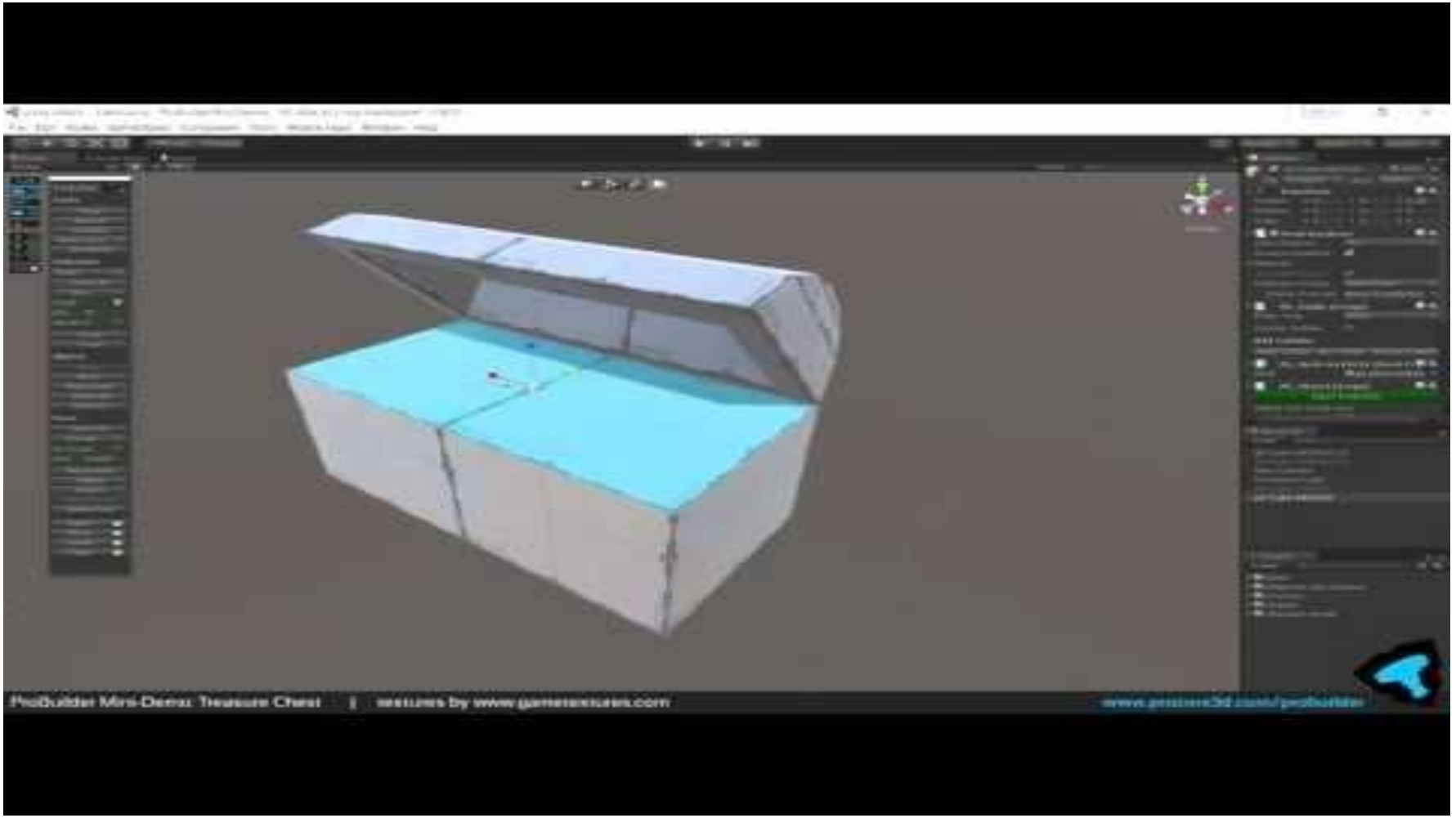


Bevel edge

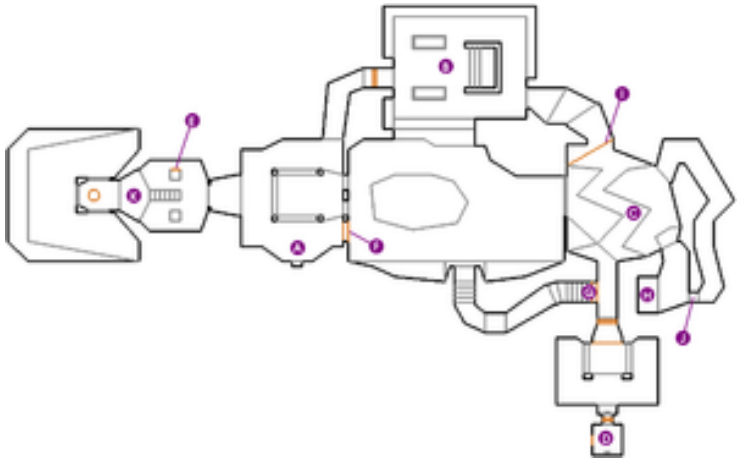
Getting Started with ProBuilder for Unity



Treasure Chest



Level Editor



PROBUILDER "*HOW TO MAKE...*" TUTORIAL SERIES:

RE-CREATING THE "E1M1" LEVEL FROM DOOM, IN UNITY 3D

PART 1: FLOORS

PART 2: WALLS

PART 3: CEILINGS

PART 4: COLORIZING

PART 5: TEXTURING

PART 6: DOORS

PART 7: CUSTOM COLLISION

PART 8: OCCLUSION

PART 9: TRIGGERS

PART 10: LIGHTING