COMP30019 Graphics and Interaction
Perspective & Polygonal Geometry

Adrian Pearce

Department of Computing and Information Systems
University of Melbourne

The University of Melbourne

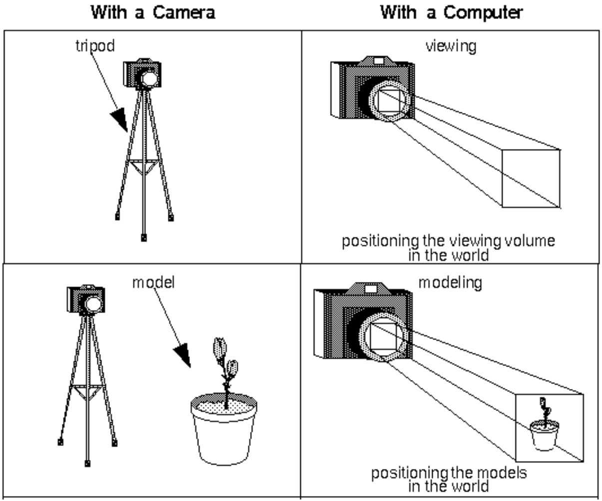
Lecture outline

Perspective geometry

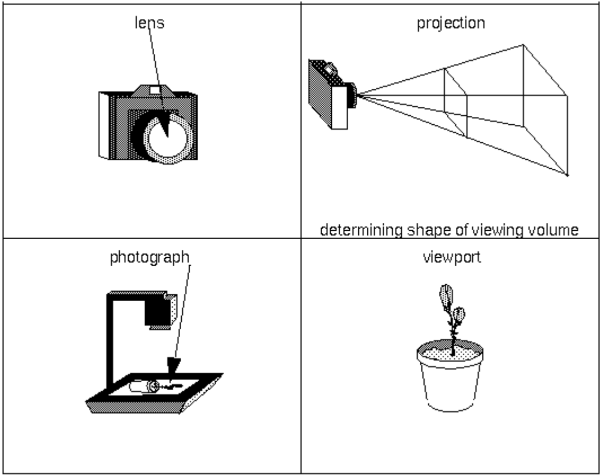
How are three-dimensional objects projected onto two-dimensional images?

Aim: understand point-of-view, projective geometry.

Viewing


With a Camera
With a Computer
tripod
viewing
positioning the viewing volume in the world
model
modeling
positioning the models in the world

Viewport

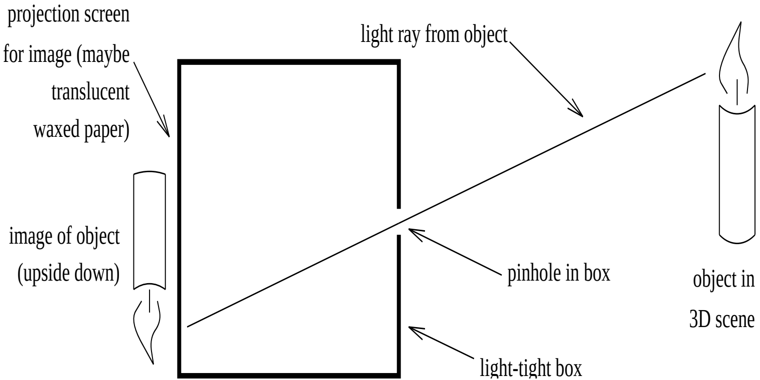

lens
projection
determining shape of viewing volume
photograph
viewport

Geometry of image formation

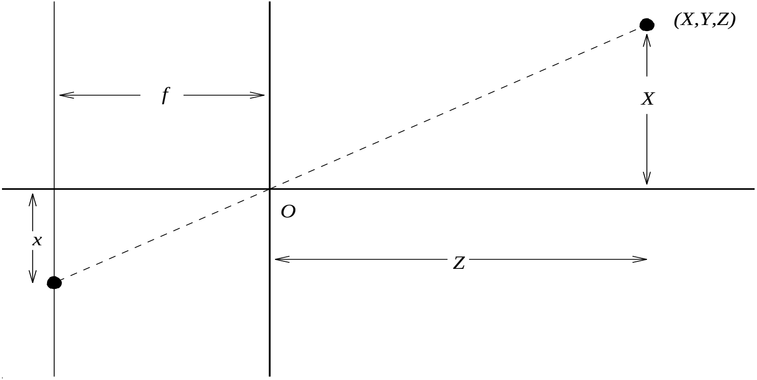
Mapping from 3D space to 2D image surface, moving from a higher dimensional image to a lower dimensional image

- The X, Y, Z points in the three dimensional world, sometimes called voxels, are transformed in to x, y pixels in a two-dimensional image.
- More specifically, a mapping from 3D directions, *rays of light*, to/from the observer.

Pinhole Camera



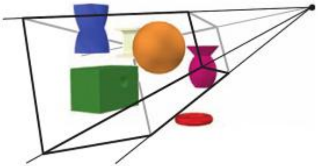
Perspective geometry

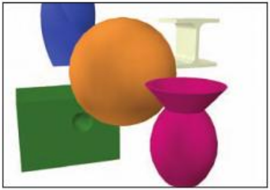


Perspective geometry

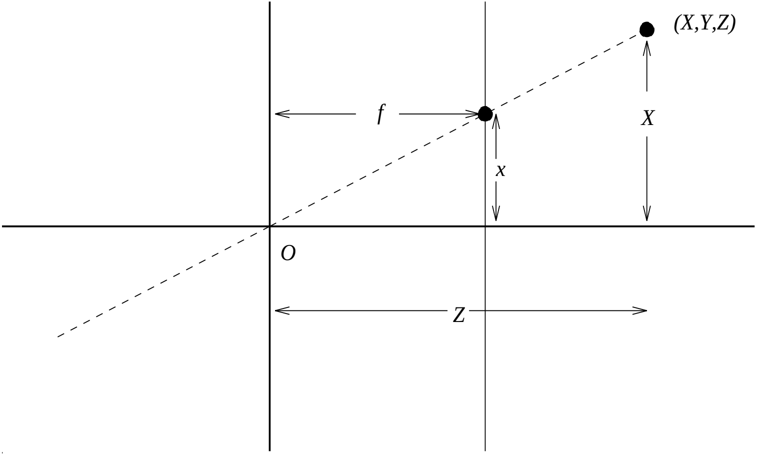
- Basically an abstraction of pin-hole camera.
- Look at XOZ plane (same thing happens in YOZ plane).
- Actual point in 3D space is (X, Y, Z)
- 0 is origin (focal point) or centre of projection.
- Z is distance from actual point to origin.
- f is focal distance (focal length).
- x is the image (upside down) with respect to real world.

Perspective of *virtual camera*





Virtual camera geometry



Virtual camera geometry

- Image projection surface imagined to be in front of projection centre.
- Geometrically equivalent
- Often more convenient to think about projection

Perspective Formulas

Point $P = (X, Y, Z)$ in 3D space has projection (x, y) in the image where

$$
\begin{aligned}
\frac{x}{f} &= \frac{X}{Z} \\
\frac{y}{f} &= \frac{Y}{Z}
\end{aligned}
$$

or

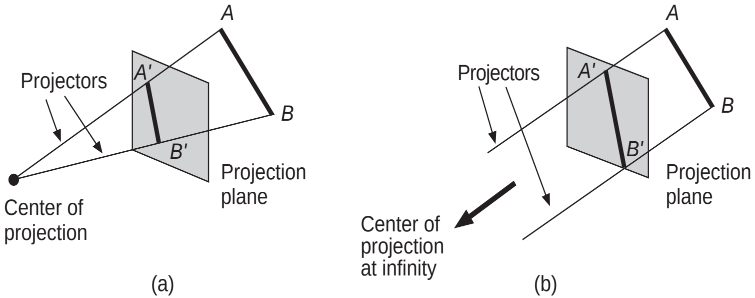
$$
\begin{aligned}
x &= \frac{Xf}{Z} \\
y &= \frac{Yf}{Z}
\end{aligned}
$$

f being the “focal distance” (sometimes f is called d).
Look at similar triangles in the previous diagram.

Perspective Formulas

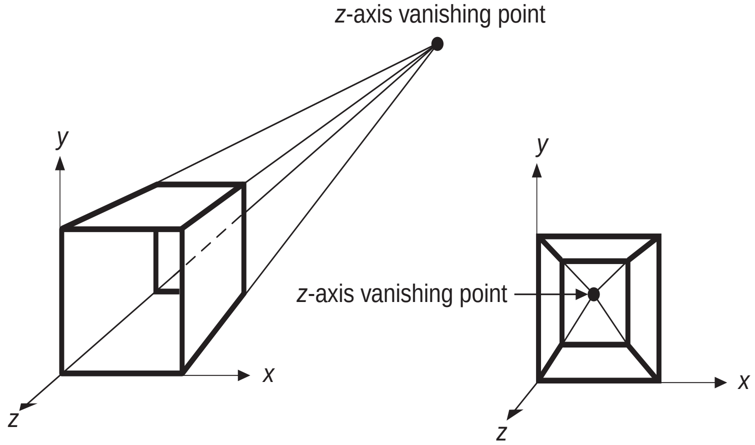
- Look at perspective projection diagram to convince yourself of this — triangles *xOf* and *XOZ* have the same proportions.
- Rearranging gives equations shown on previous slide.
- These formulas apply *only* for *camera-centred* coordinates, for which perspective projection has a particularly simple form.
- For *arbitrarily centred* coordinate systems 3D transformations are necessary (more on this when we tackle 3D transformations using matrices).

Centre of projection



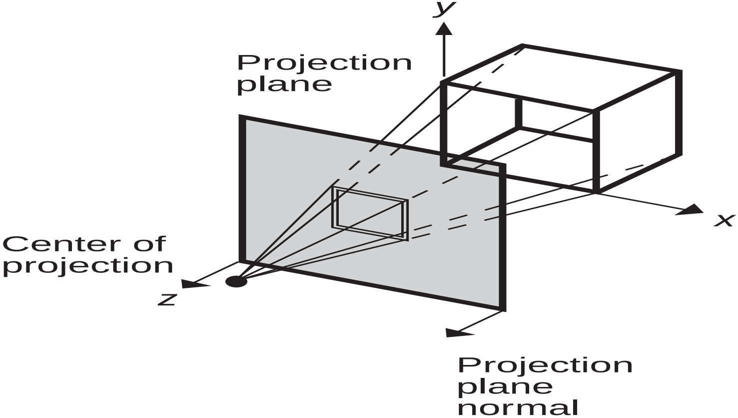
(Foley, Figure 6.03)

One point perspective projection



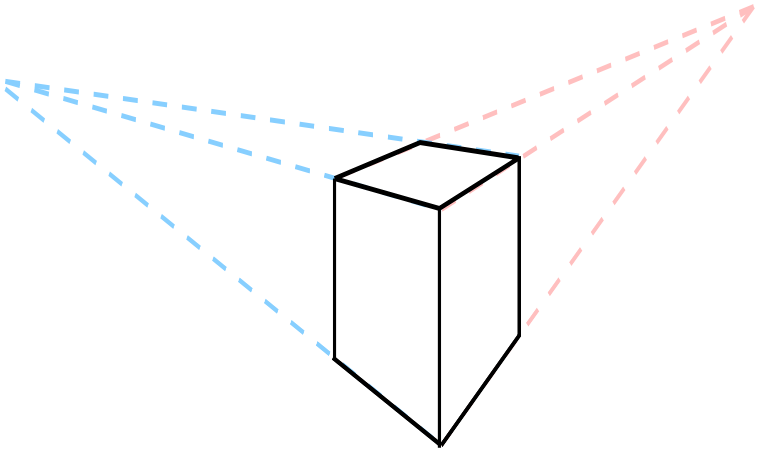
(Foley, Figure 6.04)

One-point perspective projection

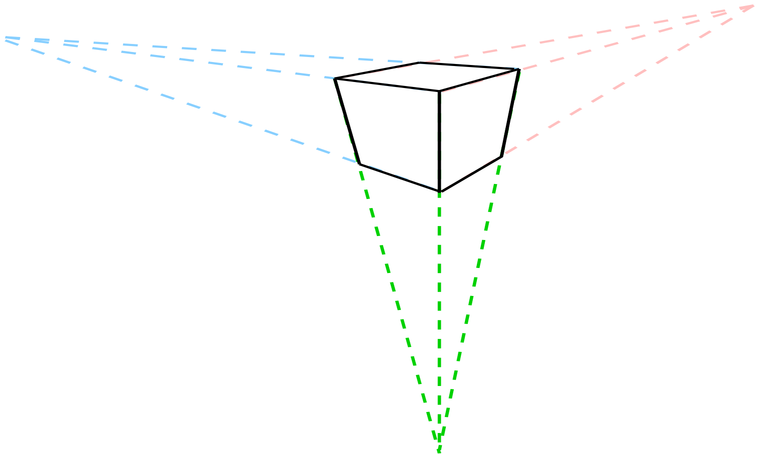


(Foley, Figure 6.05)

“Two-point” perspective



“Three-point” perspective



Vanishing points

- In 3D, parallel lines meet only at infinity, so the vanishing point can be thought of as the projection of a point at *infinity*.
- If the set of lines is parallel to one of the three principal axes, the vanishing point is called an *axis vanishing point*.
- So called “one-point”, “two-point”, and “three-point” perspectives are just special cases of perspective projection, depending on how image plane lines up with significant planes in scene.
- In fact, there are an *infinity of vanishing points*, one for each of the infinity of directions in which a line can be oriented.

Perspective of the human eye

Human eye effectively uses a kind of “spherical” projection:

Retina is curved, though projection centre (in lens) isn’t at centre of the eyeball (therefore not planar geometric projection).

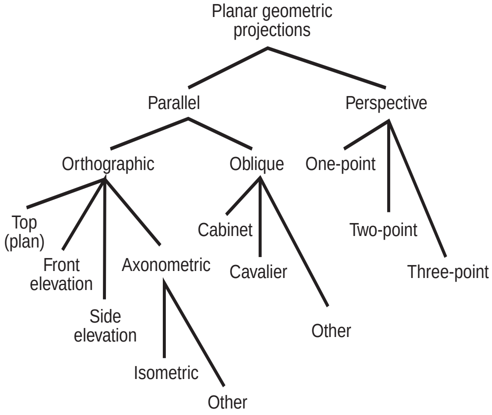
- Doesn’t exactly match perspective projection.
- Only a problem for very wide fields of view.

Perspective is basically the right projection for putting a 3D scene onto a flat surface for human viewing.

- Other projections are possible for special effects, e.g. “fish-eye” lens.

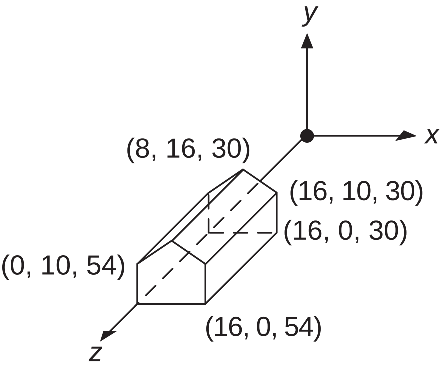
Classes of projection

Subclasses of planar geometric projections



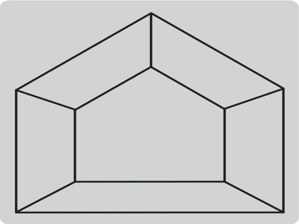
(Foley, Figures 6.21 & 6.22)

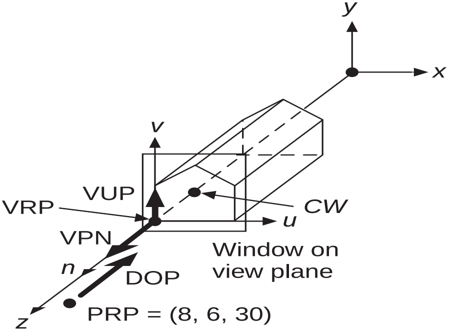
House example



(Foley, Figure 6.4)

One-point, centred perspective projection example





(Foley, Figures 6.21 & 6.22)

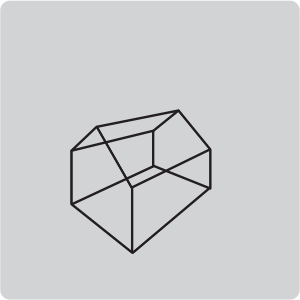
Which of the below is the centre of project in the Figure?

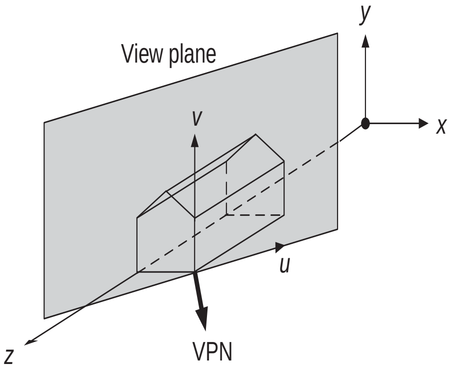
Is the view plane inbetween the centre of projection and the house or behind the centre of projection?

- VRP (view reference point)
- PRP (projection reference point)
- VPN (view plane normal)
- DOP (direction of projection)
- VUP (view-up vector)

Two-point perspective projection example

In a two-point projection of a house, left, the viewplane (defined by the view plane normal, VPN), right, *cuts* both the *z* and *x* axes





Ortographic (parallel) projection

Parallel projection (also known as *orthographic projection*) is given by

$$\begin{aligned} x &= X \\ y &= Y \end{aligned}$$

That is, just drop Z coordinate!

Perspective Geometry Summary

- Perspective geometry is based loosely on the *pin-hole* camera model that maps 3D points onto a 2D image plane
- The image plane may thought of either behind a *focal point* or in between a *vanishing point* and the object.
- Computer graphics largely concerns *planar geometric projections*, generally *perspective projection* and sometimes *parallel projection* for specific applications.
- *One-point*, *two-point* and *three-point* projection variants arise according to how many times the viewplane cuts the axis.

Perspective Geometry in Unity

Unity tutorials (getting started; perspective in Unity):

`https://unity3d.com/learn/tutorials`

Roll-a-Ball tutorial (is excellent)

`https://unity3d.com/learn/tutorials/projects/roll-ball-tutorial`

Github for Unity's official Roll-a-ball tutorial:

`https://github.com/nicksuch/Roll-a-ball`

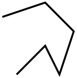
Unity manual(s):

`https://docs.unity3d.com/Manual/`

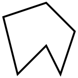
THE UNIVERSITY OF MELBOURNE

Polygonal geometry

In general a *polygon* is any plane figure bounded by straight line segments and but can comprise these forms



- polygonal arcs (polylines)



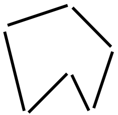
- polygonal boundaries (closed polylines)

- and filled polygons

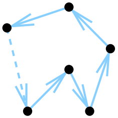
Polygons are very useful, both in themselves and as building blocks for approximating arbitrary curved arcs and regions.

Representation

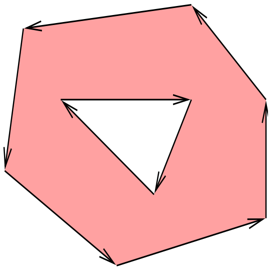
- Either as a set of line segments,



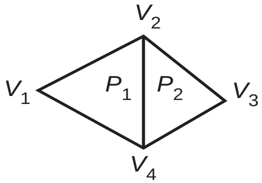
- or an ordered sequence of vertices using absolute or relative coordinates,



- Walking order convention often applies, e.g. anti-clockwise for outer and clockwise for inner



Polygon mesh: vertex list

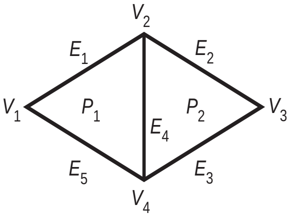


$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \ldots, (x_4, y_4, z_4))$$

$$P_1 = (1, 2, 4)$$

$$P_2 = (4, 2, 3)$$

Polygon mesh: edge list



$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \ldots, (x_4, y_4, z_4))$

$E_1 = (V_1, V_2, P_1, \lambda)$

$E_2 = (V_2, V_3, P_2, \lambda)$

$E_3 = (V_3, V_4, P_2, \lambda)$

$E_4 = (V_4, V_2, P_1, P_2)$

$E_5 = (V_4, V_1, P_1, \lambda)$

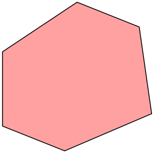
$P_1 = (E_1, E_4, E_5)$

$P_2 = (E_2, E_3, E_4)$

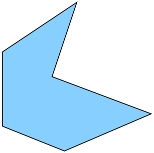
Polygons represented as line segments, *edges* can make polygon clipping and scan-line filling operations easier.

Polygon types

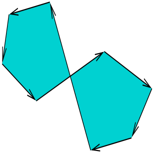
- Convex



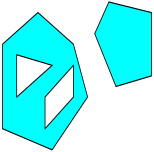
- Concave



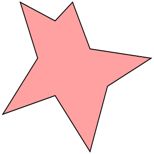
- Non-simple



- Multiple-boundary



- Star



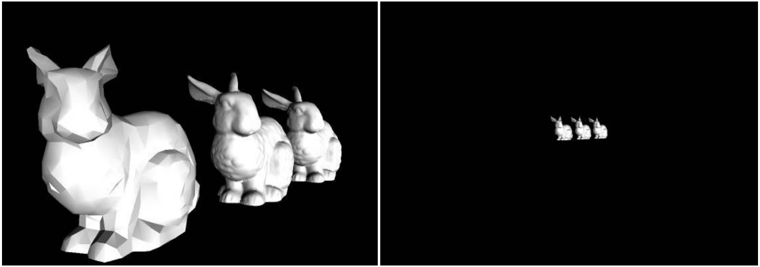
Polygon Properties

- In a *convex* polygon no internal angle is greater than 180 degrees
- In a *concave* polygon there are internal angles that can be greater than 180 degrees
- Concave polygons can be represented as a *conjunction of convex polygons* (convex polygons have certain properties that simplify geometric operations and tesselations).

Level of detail

Example from Stanford University Computer Graphics Lab

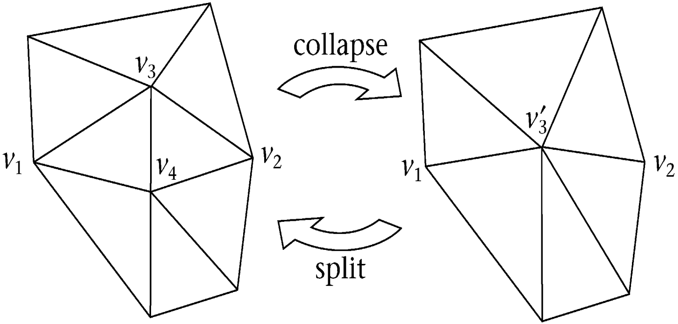
Programmer provides several different versions of an object *for viewing at different distances*.



The figures (on the left), at a distance (on the right) they appear very similar (in spite of different numbers of polygons)

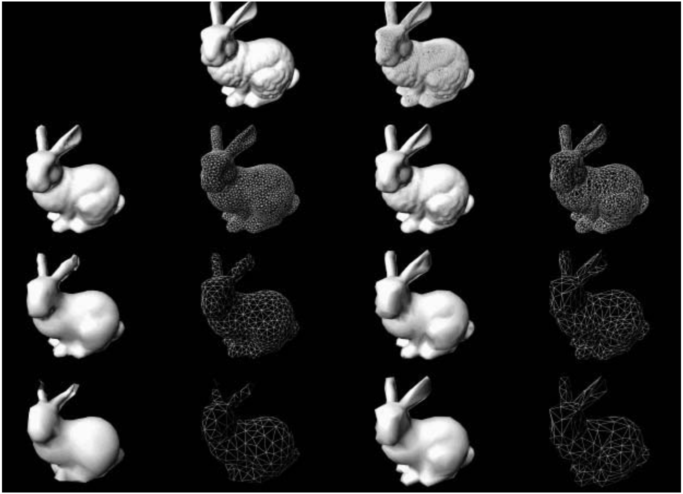
(Images from Slater text)

Vertex spit method



(Figure from Slater text)

Example of levels of detail



Polygonal Geometry Summary

- Polygons are one of the most widely used models in computer graphics and are useful for representing both two-dimensional shapes and three-dimensional objects.
- The rendering pipeline facilitates the systematic modelling, transfmation and rendering of polygons.
- Level-of-detail achieves higher performance, by adapting the number of polygons used.

Polygonal geometry in Unity & blender

- *Unity* uses the `fbx` (filmbox) file format
- If you click on models in Unity, it will launch *Model Viewer in Visual Studio*
- *blender* is more powerful and general
- *blender* uses the `blend`, file format
- You can import blender models into Unity, which invokes blenders fbx exporter