"Unity Shader Starter Essentials" book with color illustrations

---

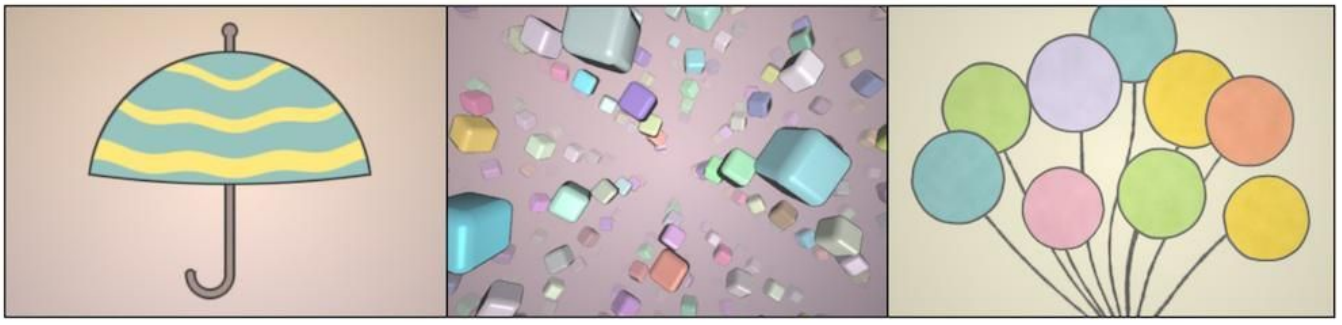
Note: This page is book "Unity Shader entry Essentials" book with color pictures Collection, the book contains all the illustrations, the

images can be searched by number when used.

Author: Feng

Lele-mail:lelefeng1992@gmail.com

Introduction



Chapter 2rendering pipeline

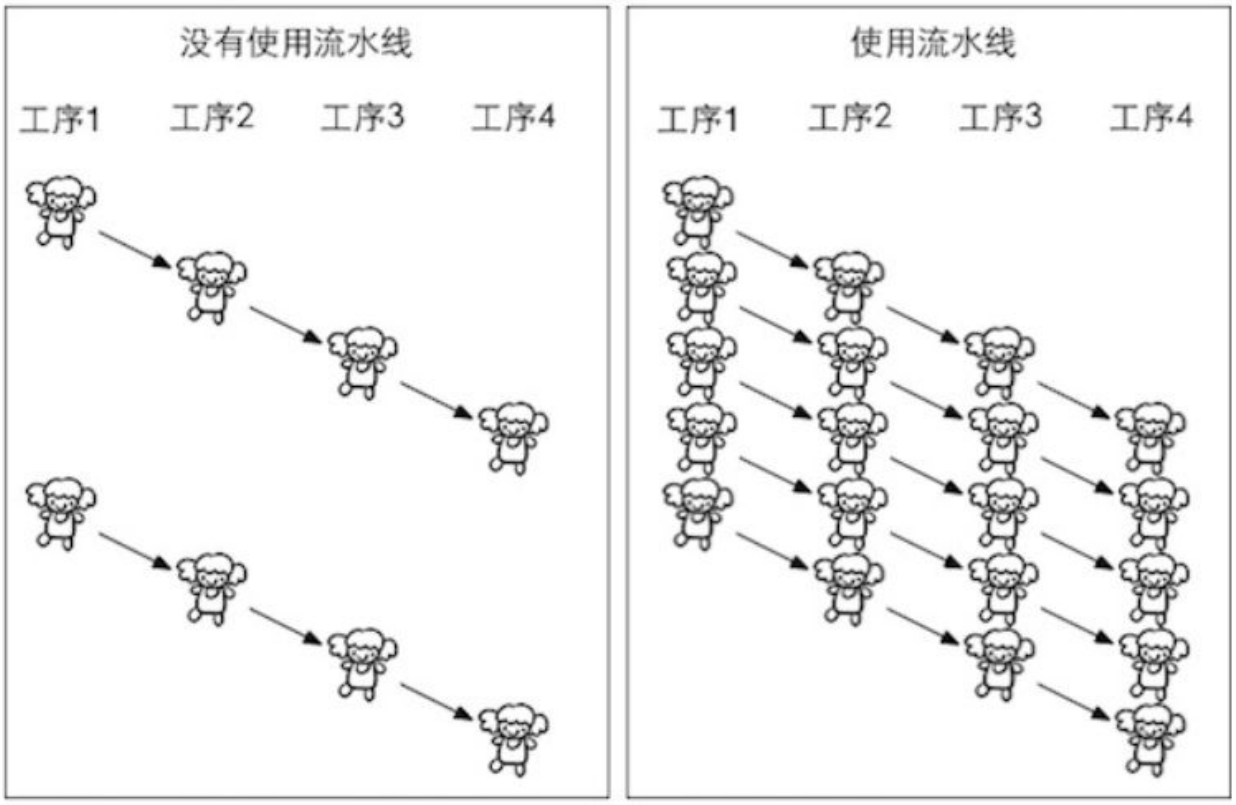


Figure 2.1 real-lifein pipeline

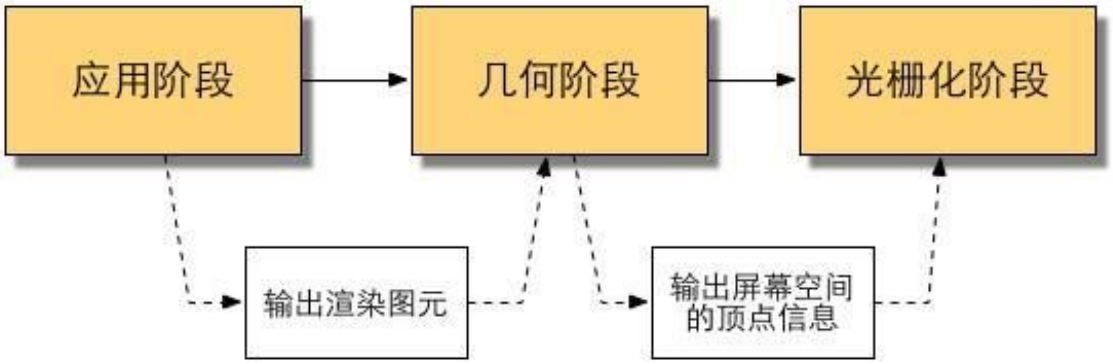
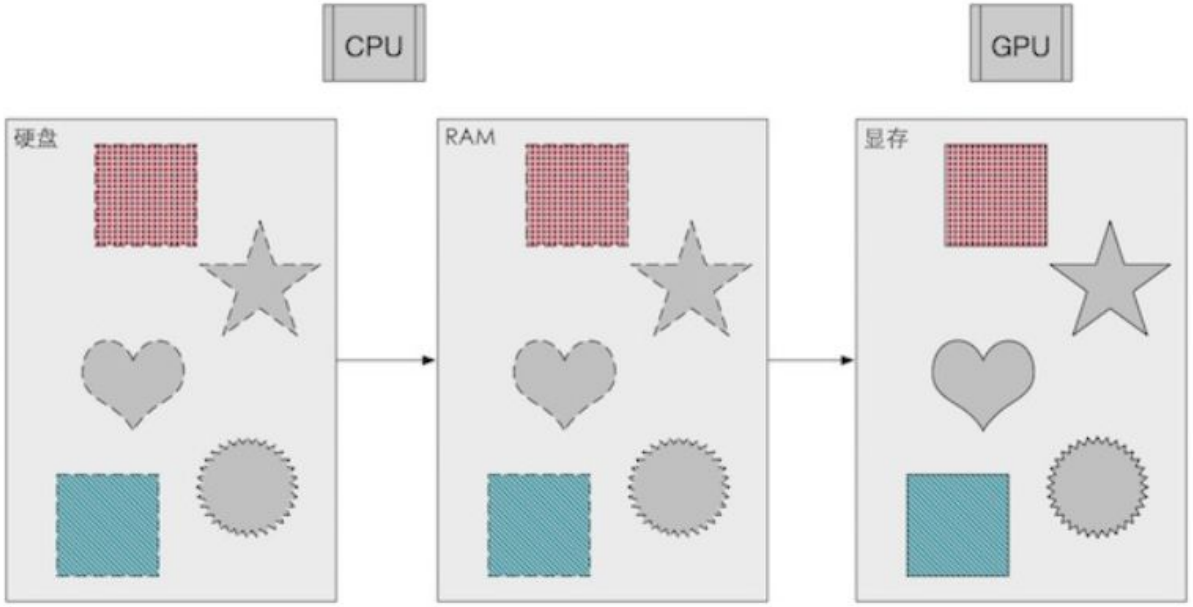
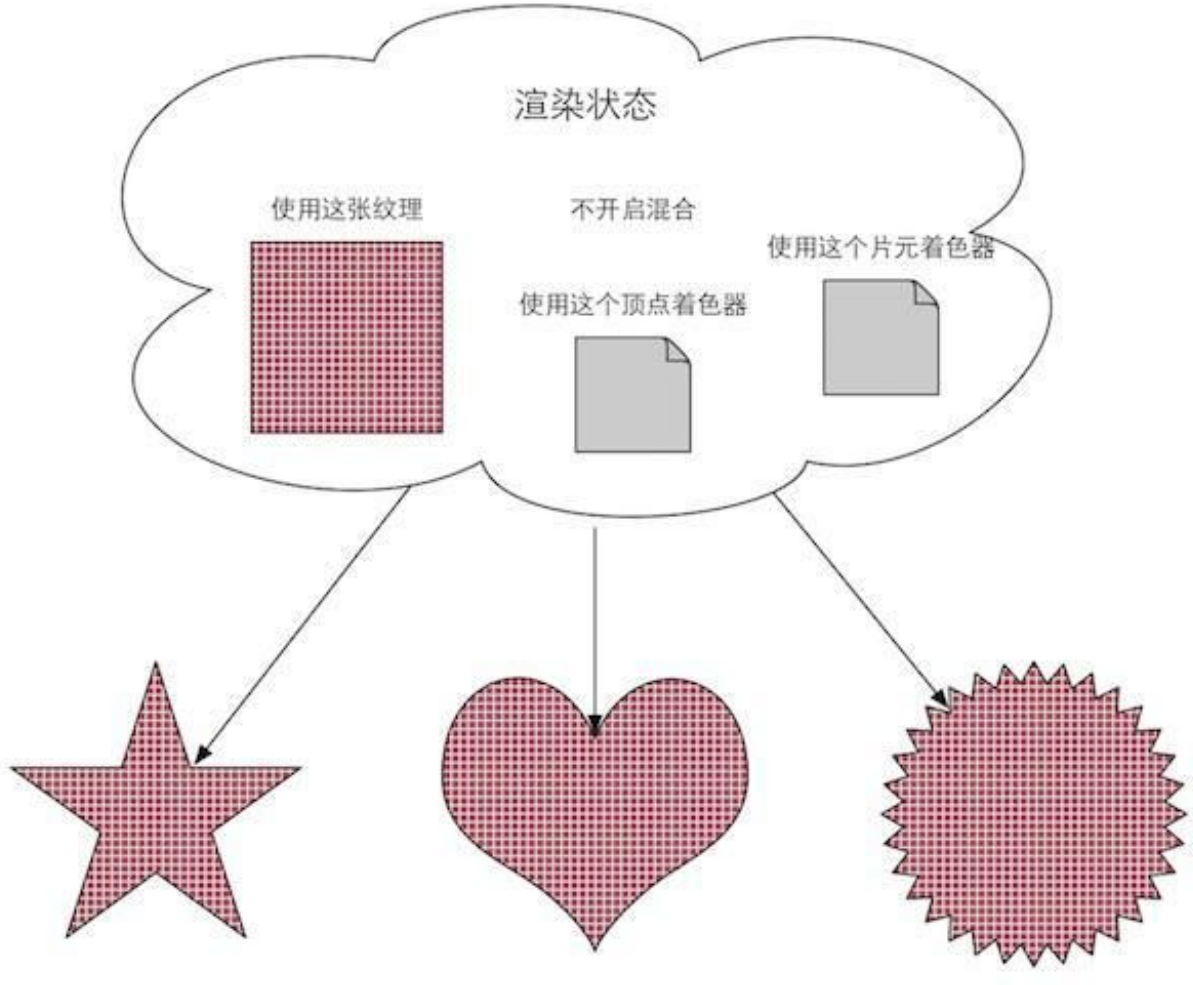


Figure 2.2renderthree conceptual phasein the pipeline



2.3the data needed to renderdiagram(two textures and three grids) the final loaded from the hard disk into video memory. When rendering, GPU can quickly access the data



in Figure 2.4 rendered three grids in the same state. Since no rendering state changes, so the appearance of the grid looks like three objects of the same material.

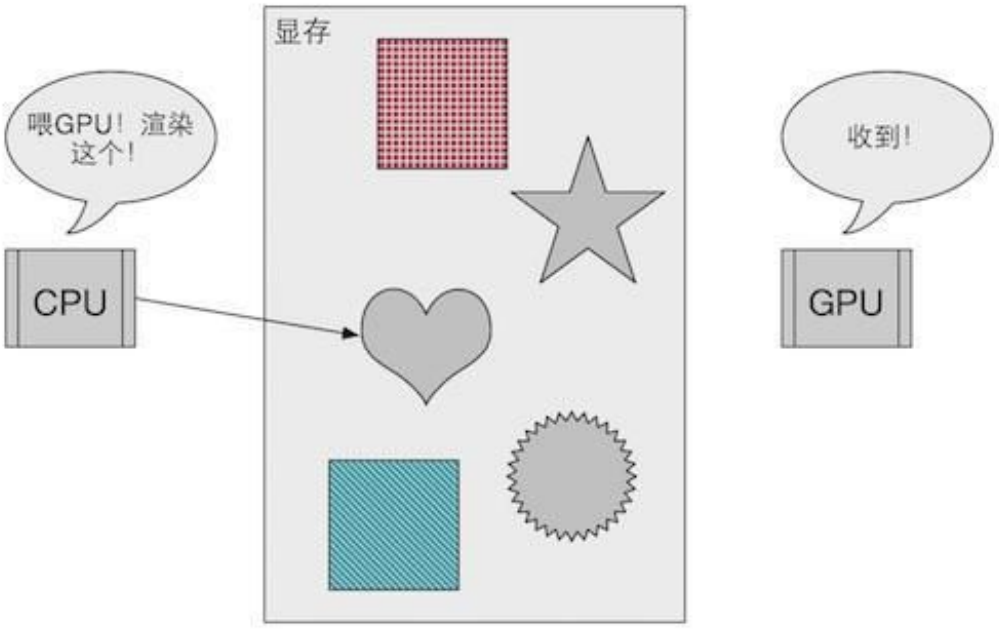
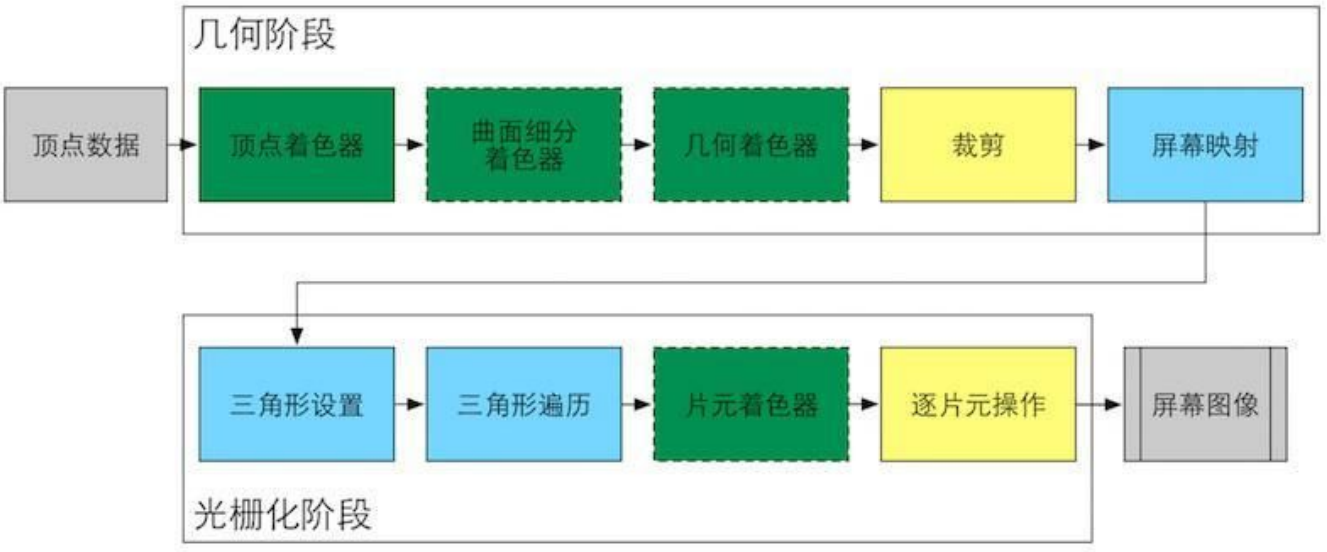


Figure 2.5 CPU to GPU told by calling the Draw Call to start a rendering process. A Draw Call will point to this call to be rendered primitivelist



rendering pipelineFigure 2.6 GPU implementation. Color indicates the programmability or configuration may be different stages: stage green indicates that the pipeline is fully programmable control, yellow indicates that the pipeline stages may be configured, but not programmable, blue indicates that the fixed pipeline stage is implemented by the GPU the developer has no control over. The solid line indicates the shader programming must be implemented by developers, a broken line indicates the optionalShader

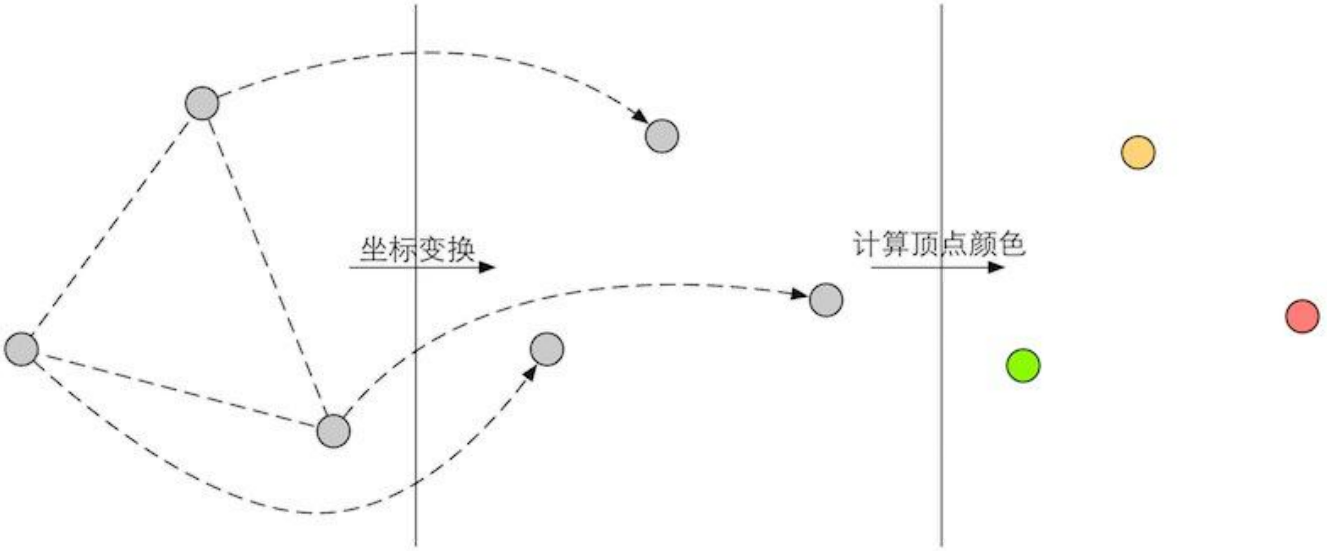
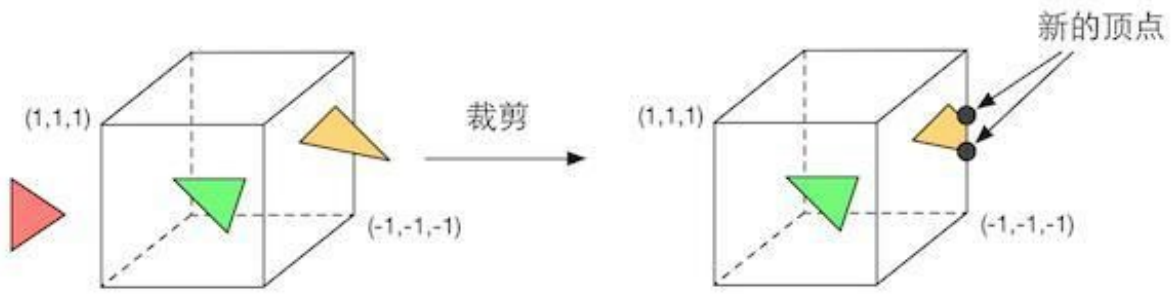


FIG2.7 GPU vertex shader is invoked on each input of the mesh vertices. Vertex shader vertex coordinate conversion must be performed, and the output color may be calculated vertices will also be required. For example, we may need to bevertex-vertexillumination,

transformed at a position 2.8 Homogeneous cut vertex coordinate space model will be shader FIG and then outputs the division made by a hardware perspective view of the obtainedNDC



graphonly in a unit cube2.9 the only entity that needs to be continued. Thus, the unit is completely outside the cube elements (red triangle) is discarded, it is completely retained inside the unit cube primitives (green triangles). And a unit cube intersecting primitives (yellow triangle) is cut, the new vertices are generated outside the original vertices are discarded

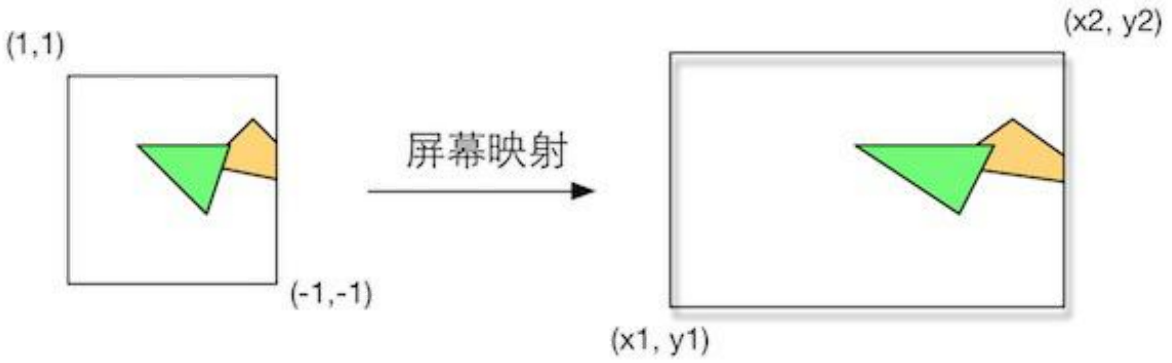


FIG screen mapping 2.10 to x, y coordinate transformation from the (-1, 1) to the range of the screencoordinates

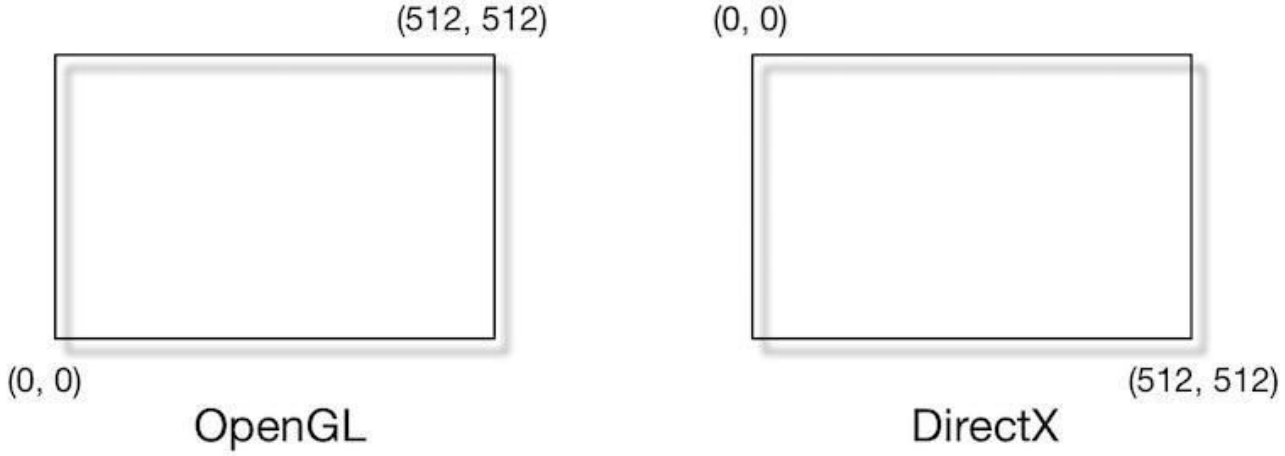
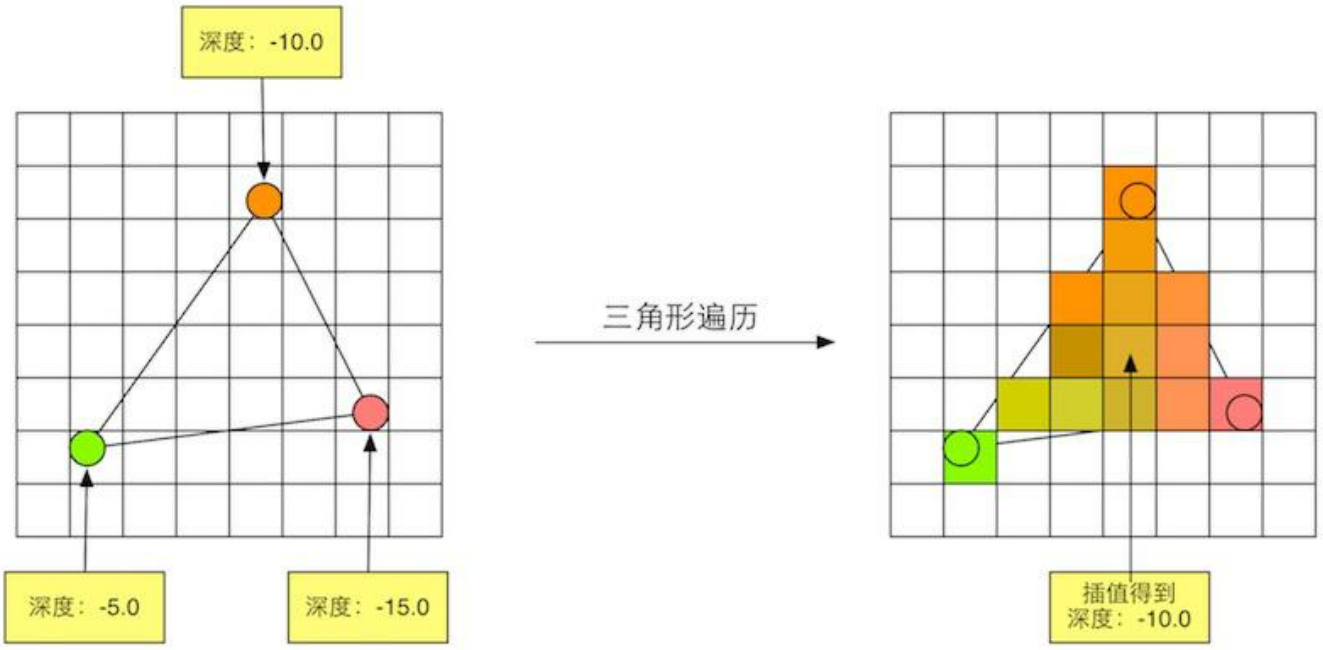


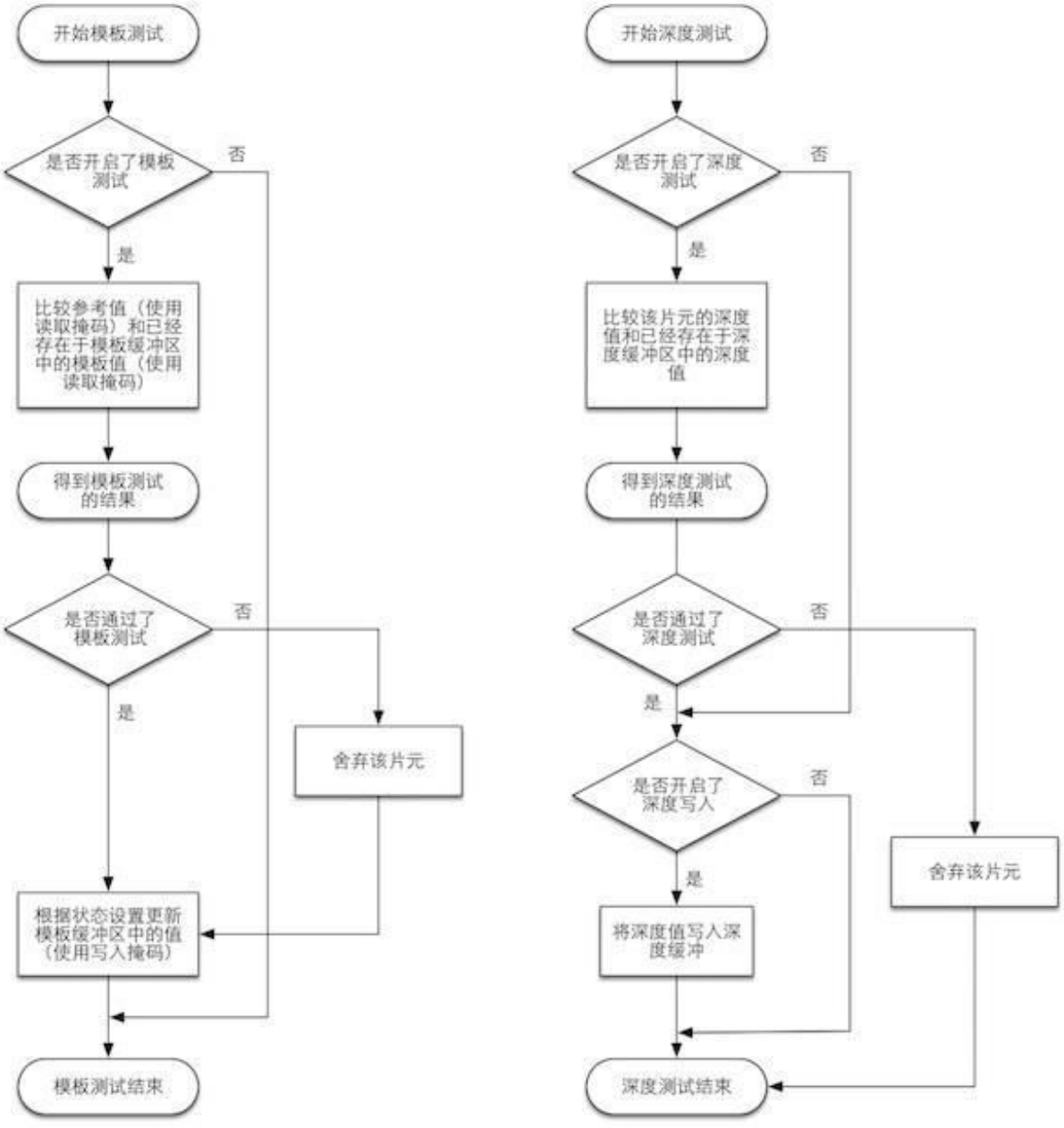
FIG2.11 OpenGL and DirectX difference in the screen coordinate system. For an image of size 512 * 512, which in OpenGL (0, 0) point in the lower left corner, and in its DirectX (0, 0) pointthetop-left corner



during of FIG triangle traversal 2.12. The vertex information output from the geometric phase, to give a final pixel position of the triangular mesh coverage. Generates a corresponding pixel fragments, fragments and state information of the three vertices is obtained by interpolation. For example, the depth of the three vertices of 2.12 obtained by interpolating the center of gravity of the sheet element corresponding to the positiondepth value -10.0

of FIG2.13 yuan information on the interpolation step according fragment shader output color calculation sheet element

2.14 done by operation of FIG fragment operations stage. Only after all the tests pass, new pixel color generation and color buffer to the fragment already present mixed,incolor bufferendthewrite



a simplified flow diagramFIGand a depth of 2.15 stencil testof testoperations.

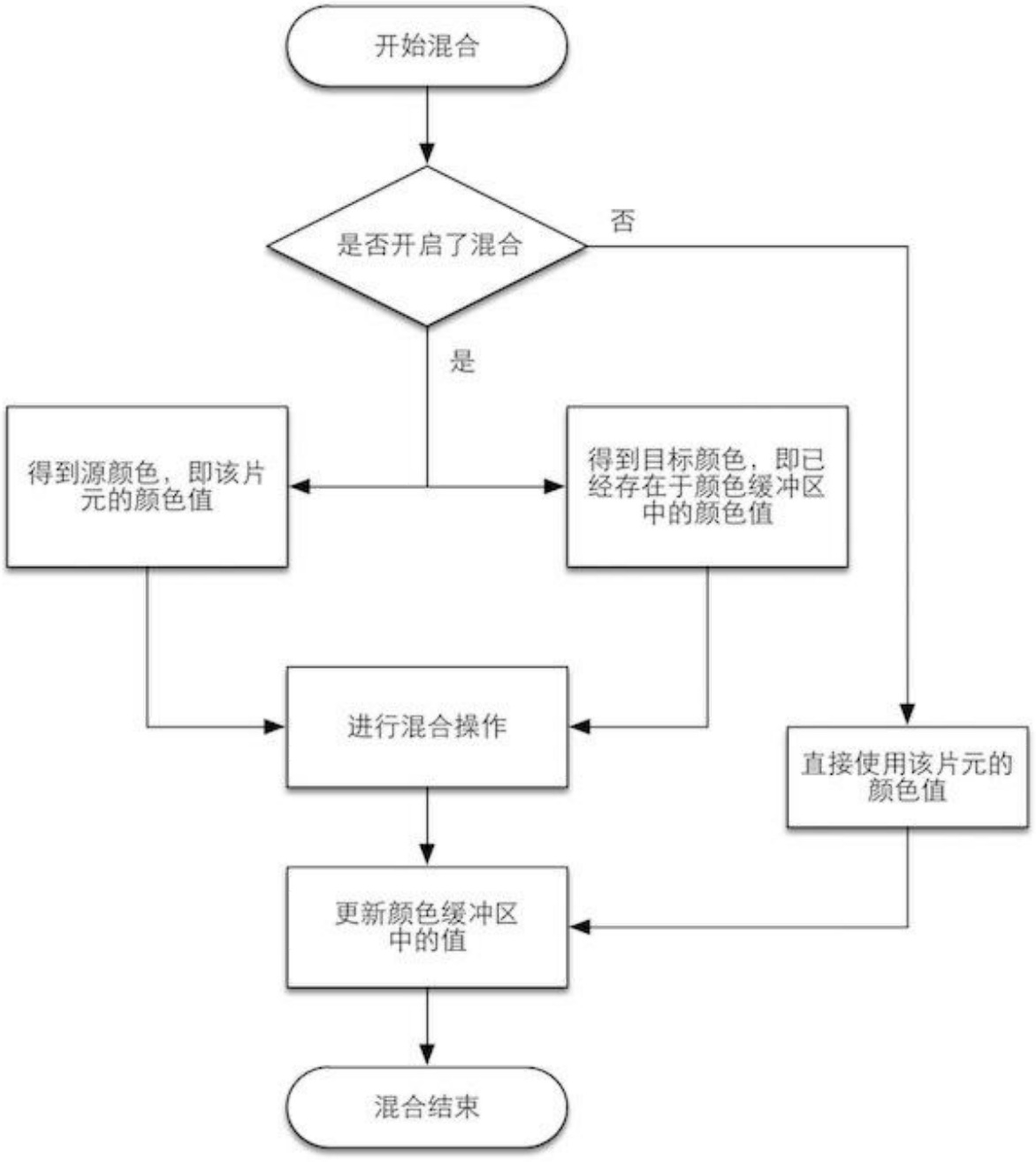
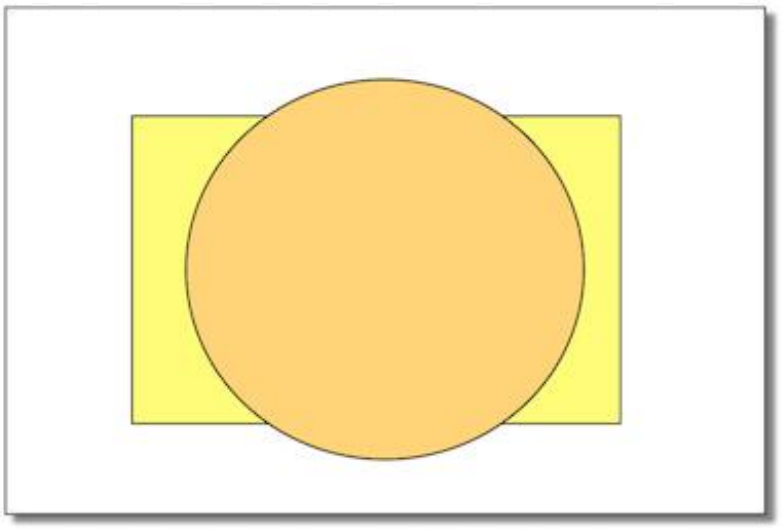
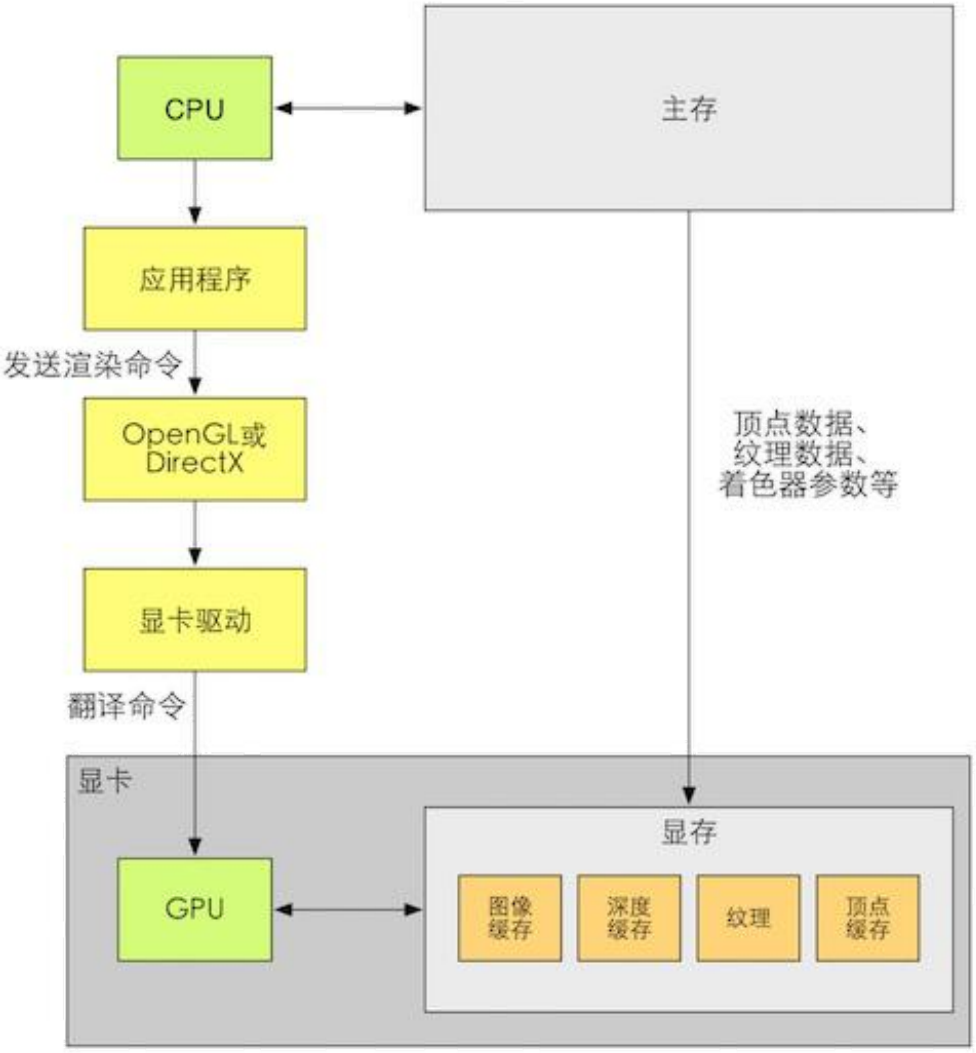
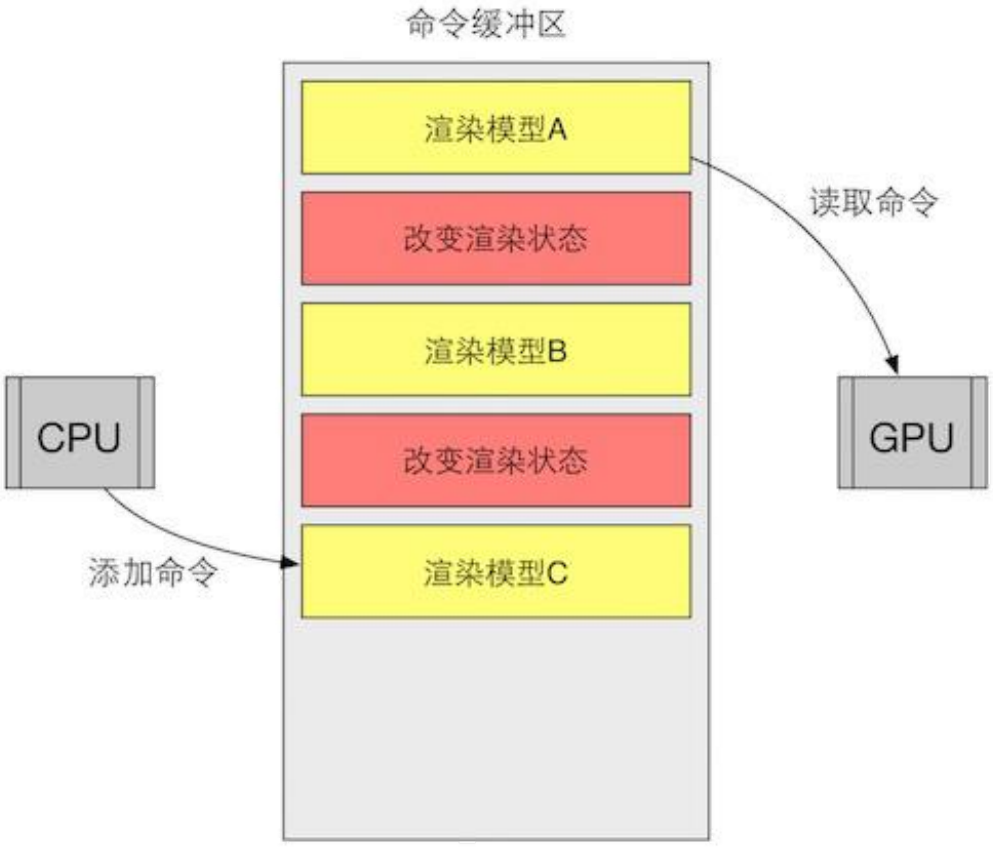


FIG simplified flowchart 2.16mixing operation

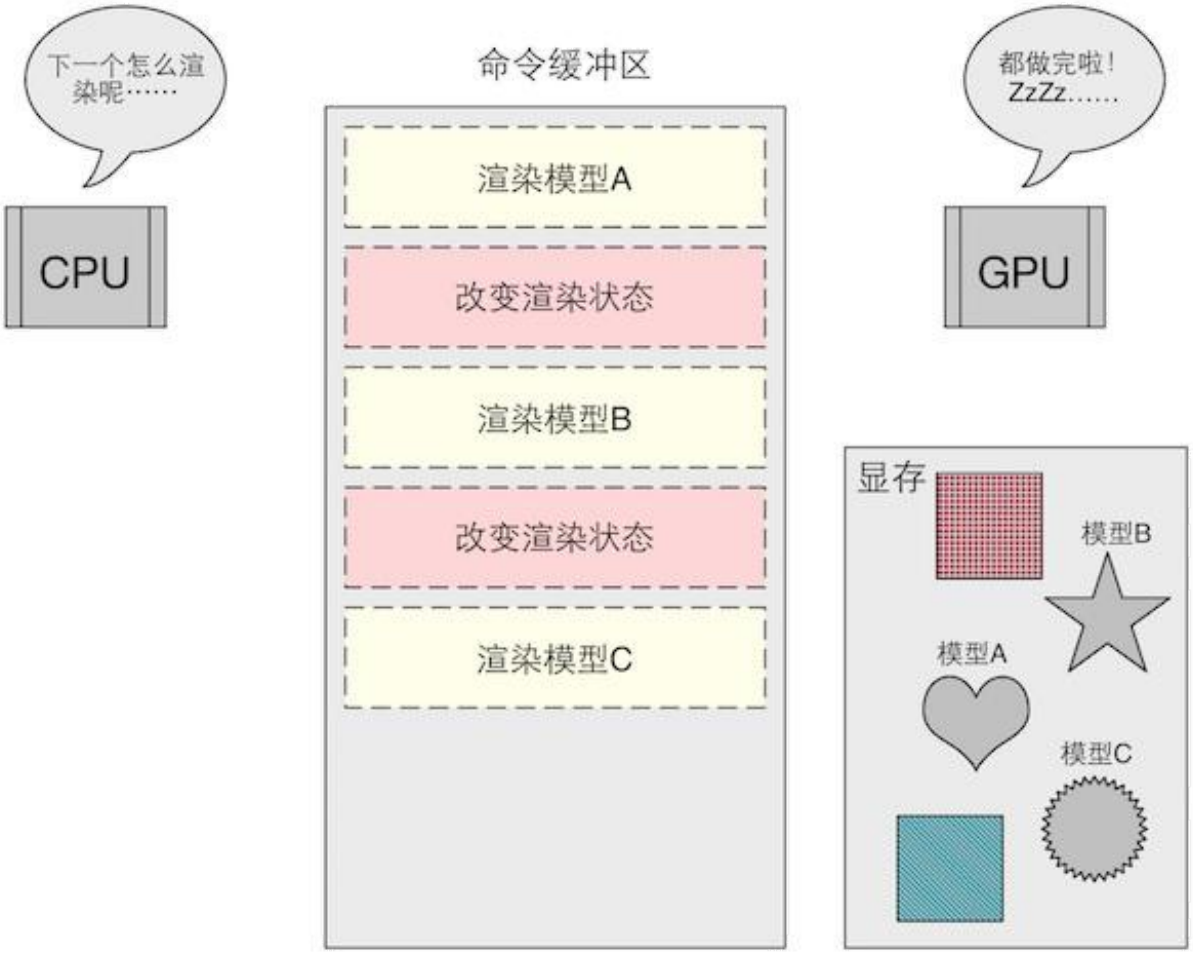
2.17illustrated  in FIG scene contains two objects: sphere and a rectangular parallelepiped, the drawing order is to draw a sphere (a circle displayed on the screen), then draw a rectangular parallelepiped (rectangular on the screen) . If the depth test is performed after the fragment shader, so when rendering cuboid, although most of its areas are obscured behind the ball, that it covers most of the fragments can not pass the depth test, but we still fragment shader need to perform these fragments, resulting in theperformance of a large waste of



between theFIG. 2.18 CPU, OpenGL / DirectX, graphics driver and GPU



relationship2.19 command buffer. CPU command added to the image programming interface command buffer, and the read command from GPU and executed. Command is in the yellow box Draw Call, and the command for changing the block in the red rendering state. We use the red square to represent the rendering state change command, because these commands tend to be more time-consuming



command Figure 2.20 dashed box indicates that the command buffer GPU has been completed. At this point, there is no command in the command buffer can be performed up, GPU is idle, and CPU are not ready for the next rendering command.

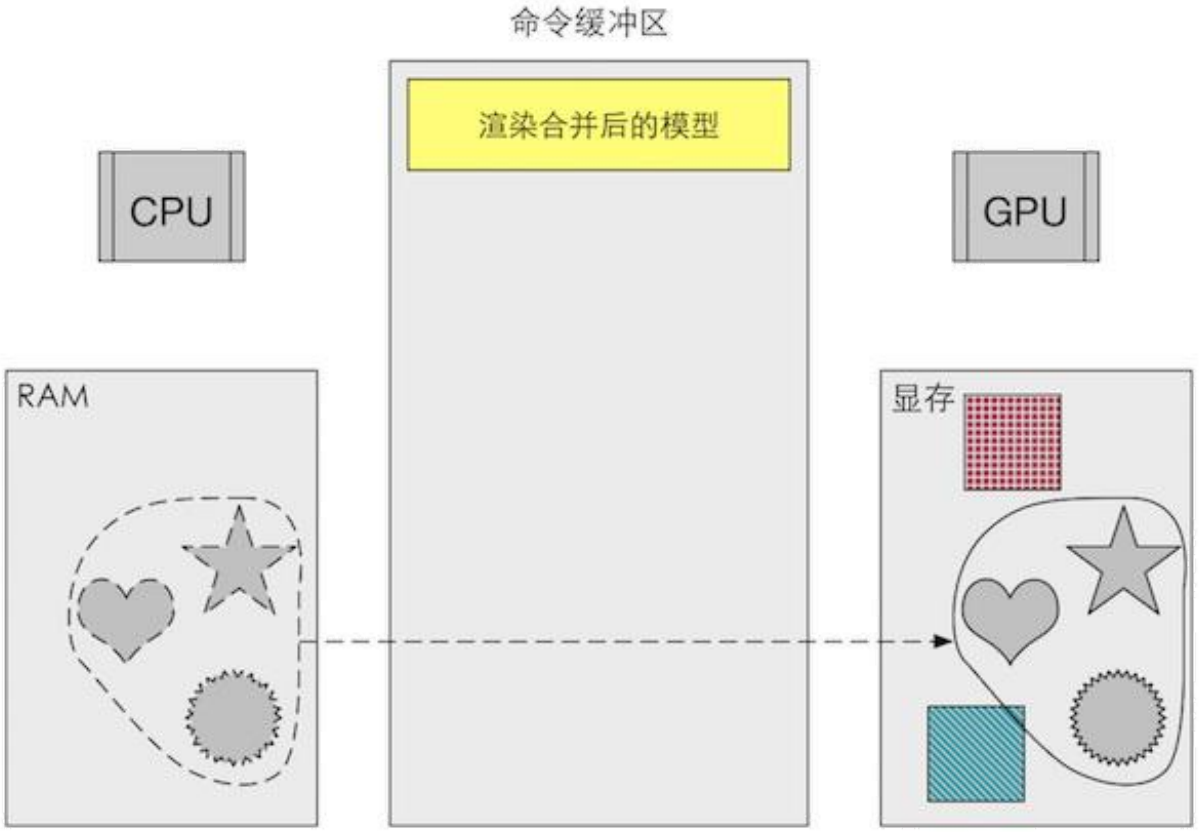
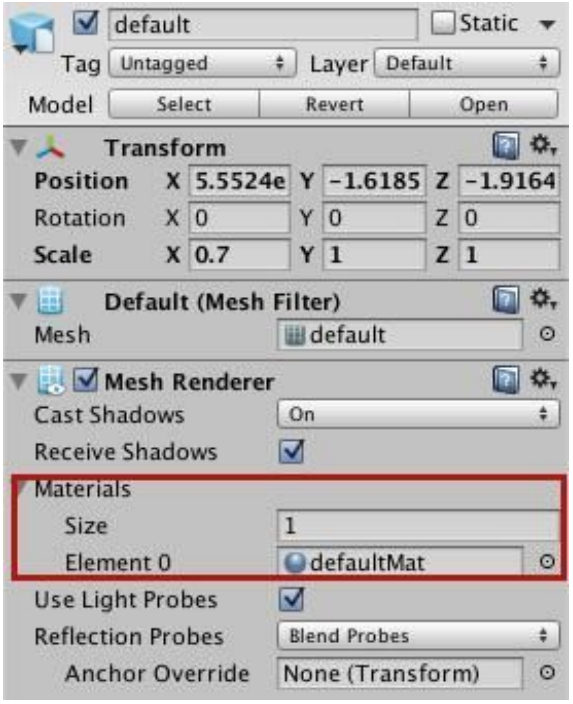


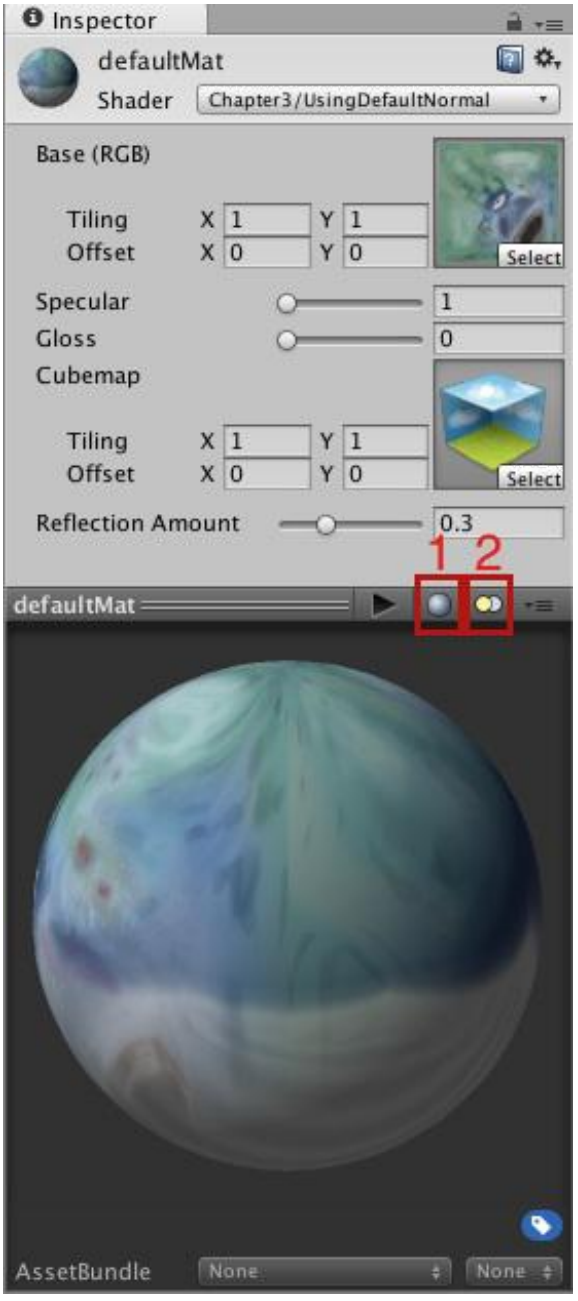
FIG. 2.21 using a batch, the RAM in the CPU are combined into a plurality of larger mesh grid, and then sent to the GPU, and then render them in a Draw Call. But it notes that the use of batch merge grid will use the same rendering state. In other words, if you need to use a different rendering state between the grid, then you can not use batch processing technology

Chapter 3 Unity Shader base

map 3.1 Unity Shader and materials. First create Unity Shader and materials required, and then assign the Unity Shader material, and adjust the attributes (such as the use of the texture, albedo, etc.) on the panel material. Finally, the material to the final view corresponding model



renderings3.2 dragged into the material of the component model Mesh Renderer



3.3 material provides a visual way to adjust the parameters used inshader

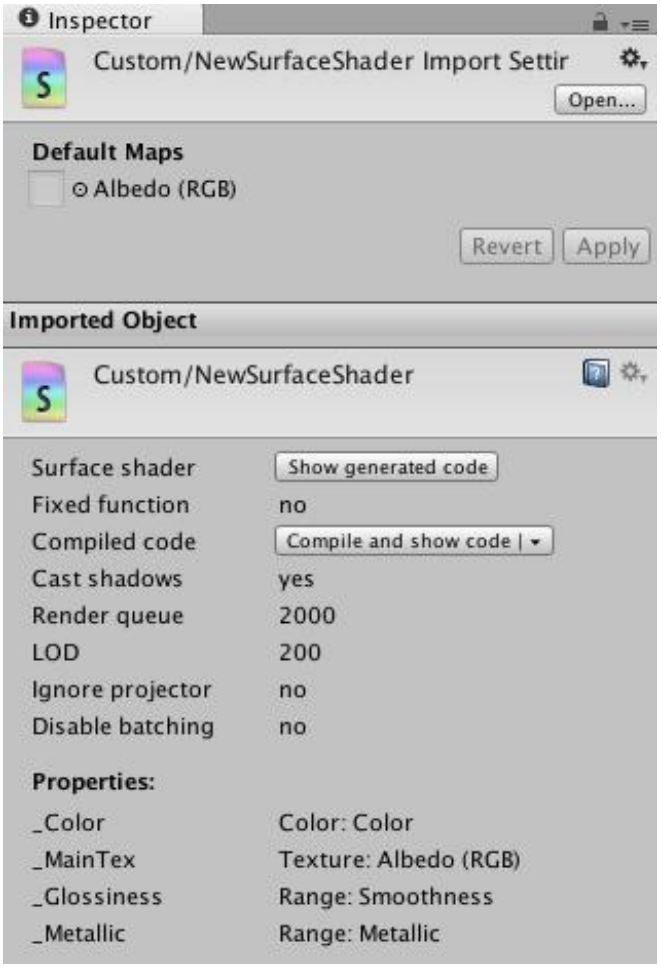


FIG3.4 Unity Shaderintroducing panel provided

FIG3.5 Gompile and show code dropdown

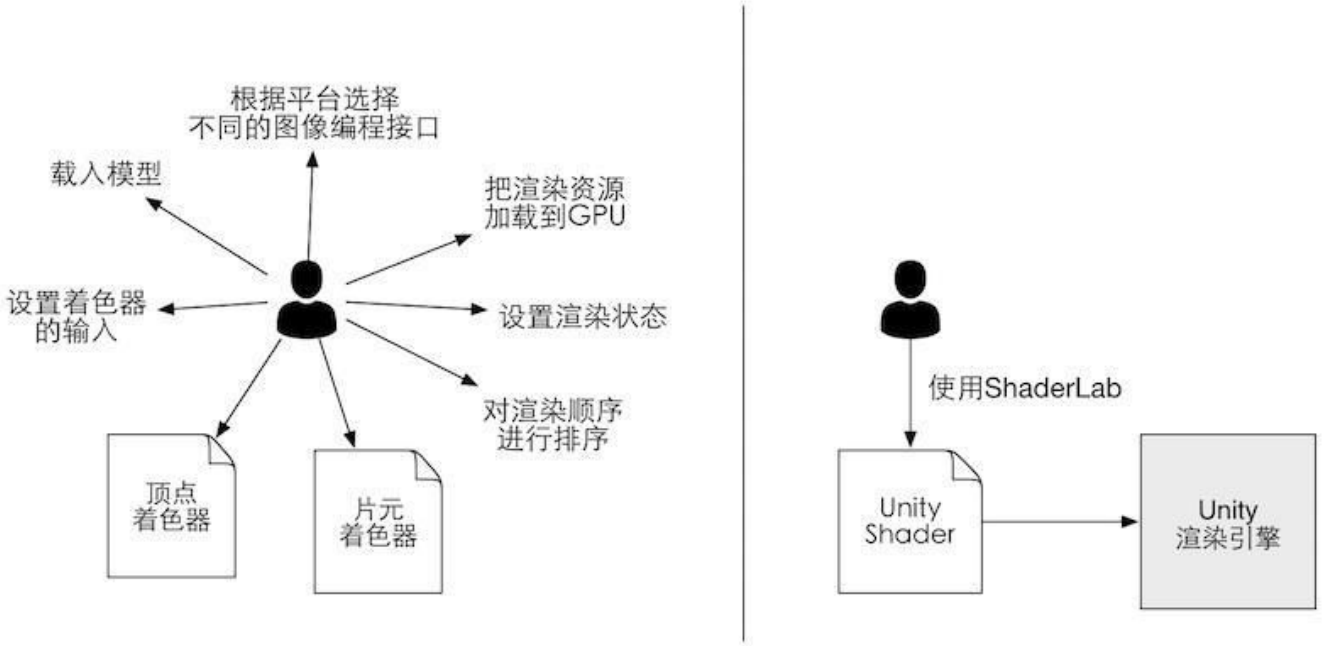
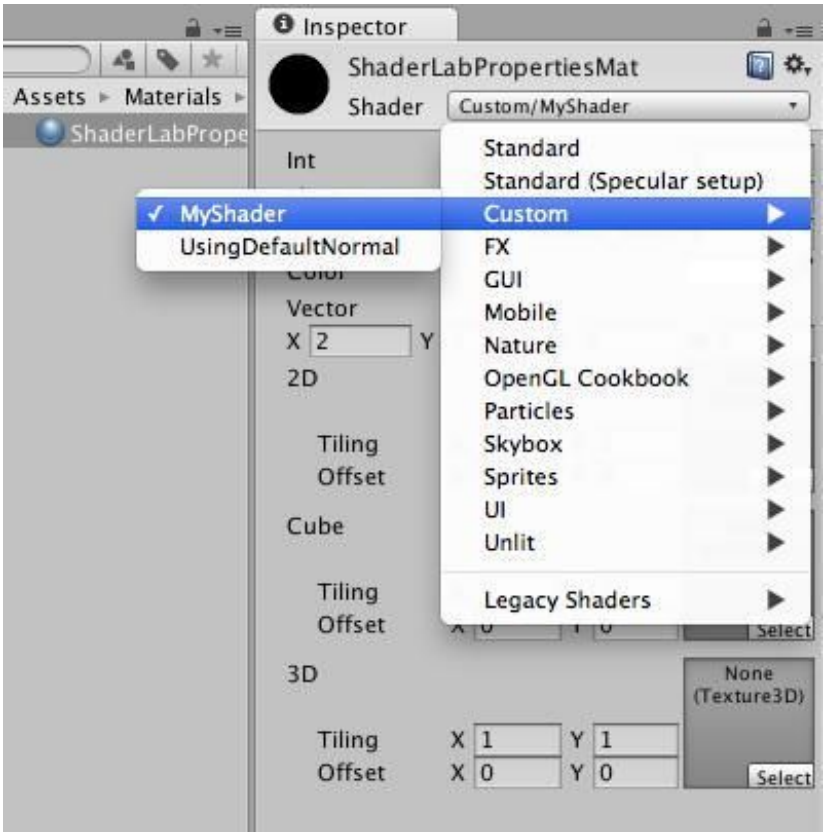
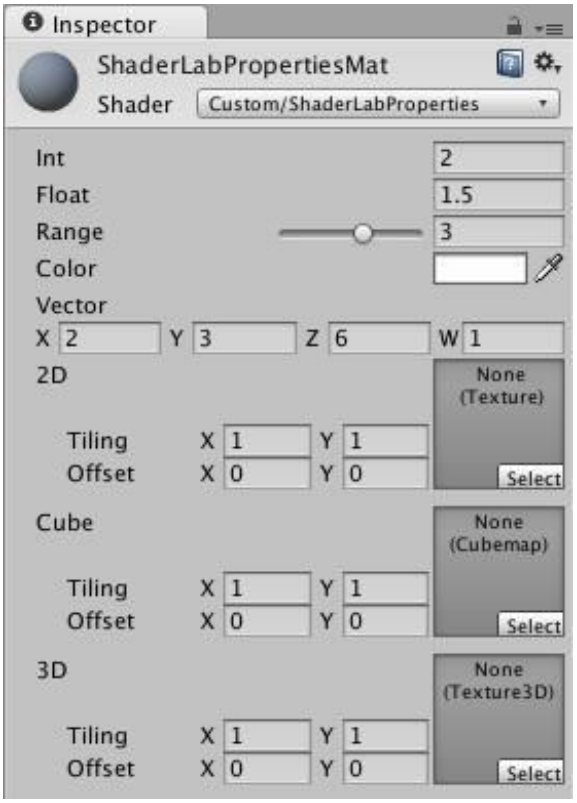


FIG 3.6 Unity Shader provides a layer of abstraction for the control of the rendering process. If you do not use Unity Shader (left), and developers need to deal with a lot of files and

settings to make the screen showing the desired effect; and with the help of Unity Shader (right), developers only need to write ShaderLab Unity Shader file can be all done



in Figure 3.7 using a slash in the Unity Shader defined name in the tissue to position the panel material



of different attributes Figure 3.8 in the display panel materialthe results

introduced intoprovided in Figure 3.9 of the panel Unity Shader Unity can view thebycompile and show code

code compiled the CG fragmentbutton.It can open the drop-down menu by clicking the inverted triangle Compile and show code button at the right end, in the drop-down menu you can select the type of platform compiler, such as assembly code only for the current graphics device compiler-specific assembly code or compiled for all platforms we also can customize the compiler to choose the platform on which

to learn the mathematical basis of Chapter 4Shaderrequired for

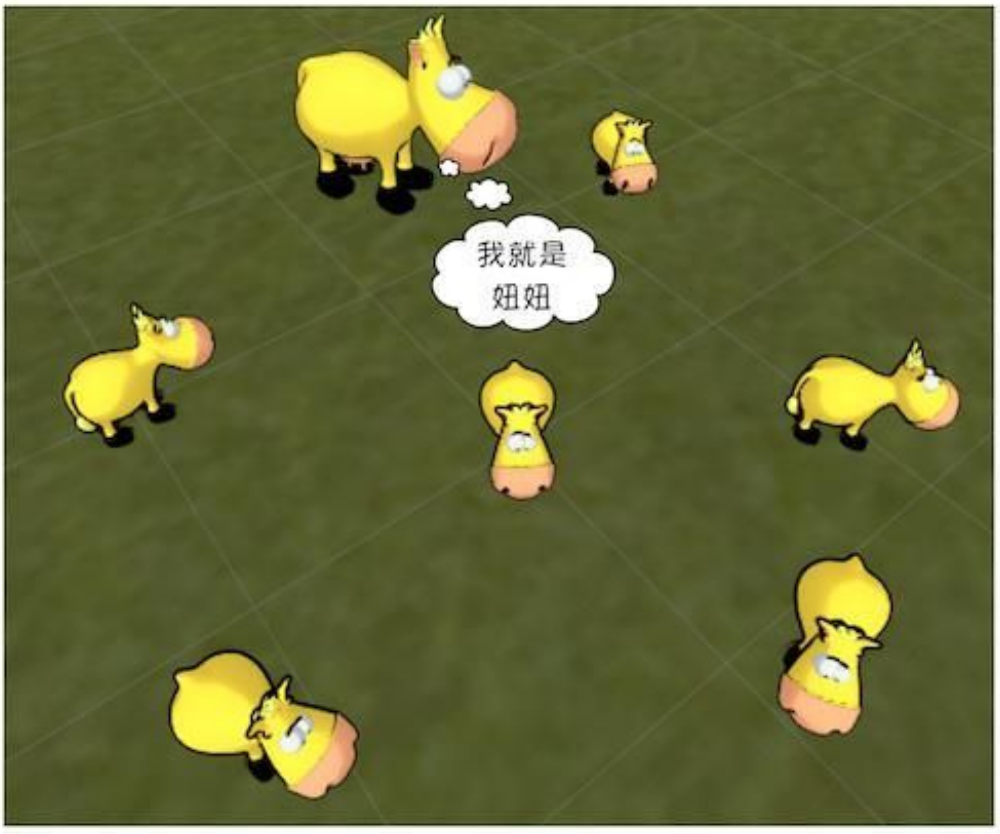
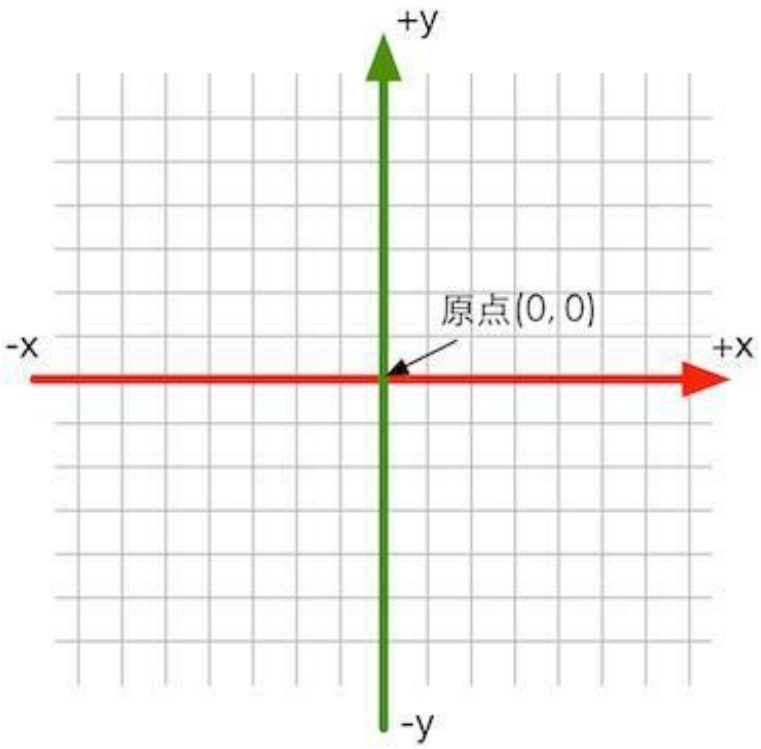


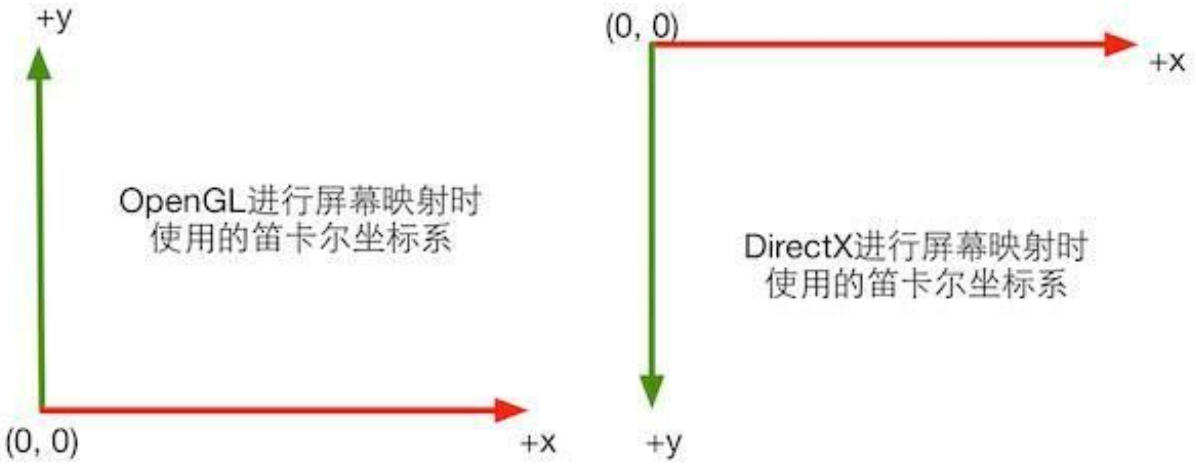
Figure4.1our game farm. Our protagonist is a niu looks the most strong, very curious cows

legend Figure 4.2, the Cartesian coordinate system Cartesian observation of a fly on the ceiling of the trajectory comes from. Descartes that

foundthe flies may be used in different distance from the wall to its current location described



in FIG. 4.3 a two-dimensional Cartesian coordinate system



in Figure 4.4 the screen mapping, OpenGL and DirectX are used in different directions of the two-dimensional Cartesian coordinate system

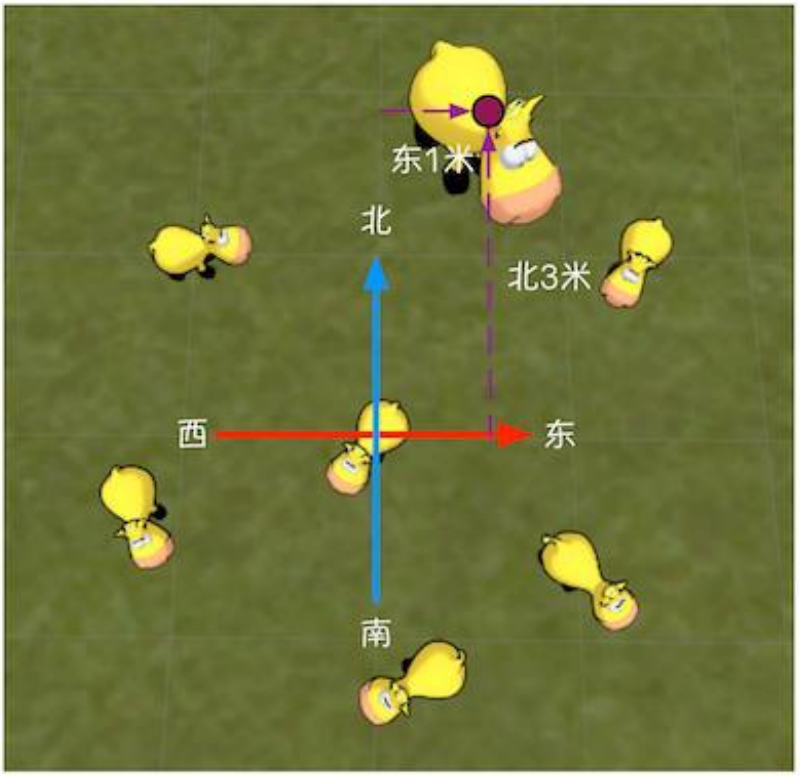
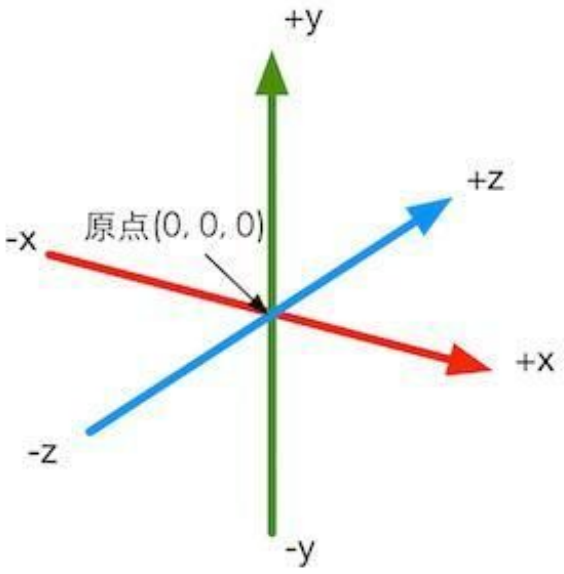


Figure 4.5 Cartesian coordinate system allows precise niuexpress their position



FIGa three-dimensional Cartesian coordinate system 4.6

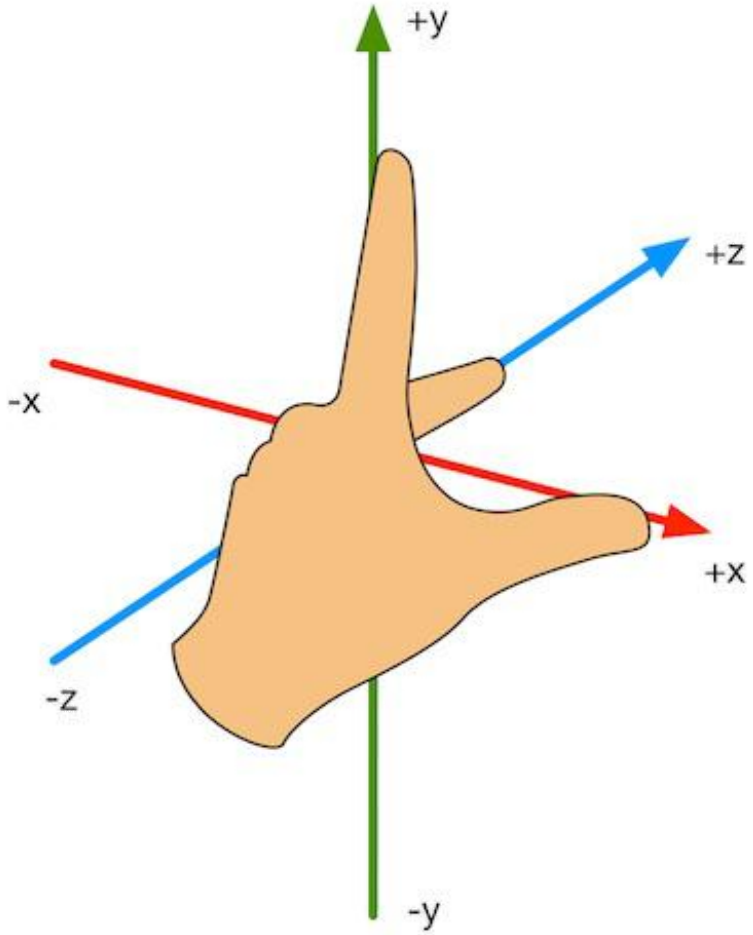


Figure 4.7left-handed

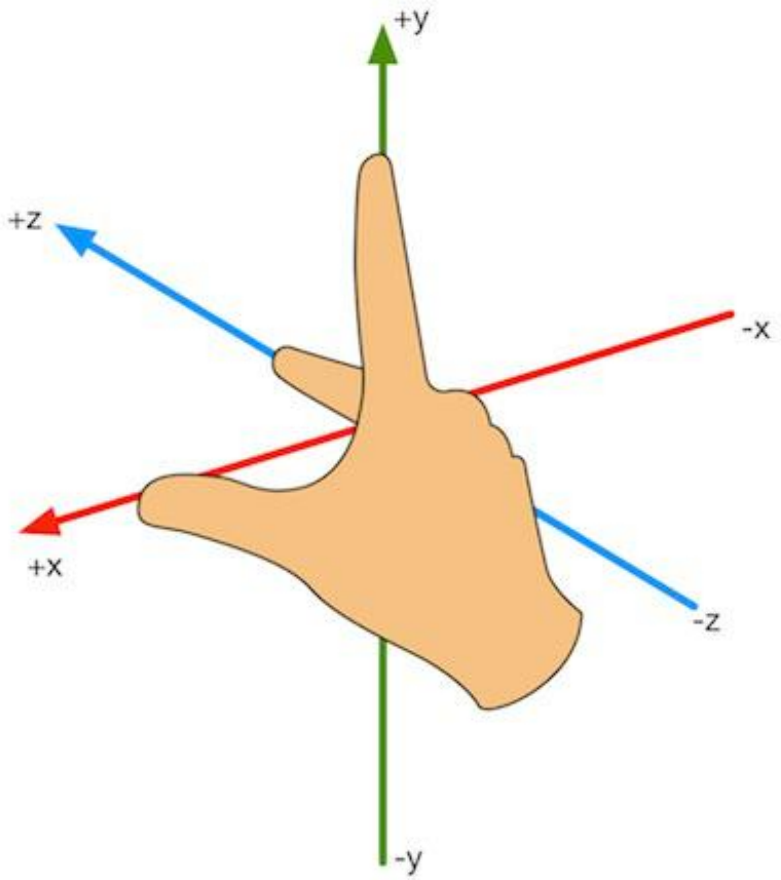
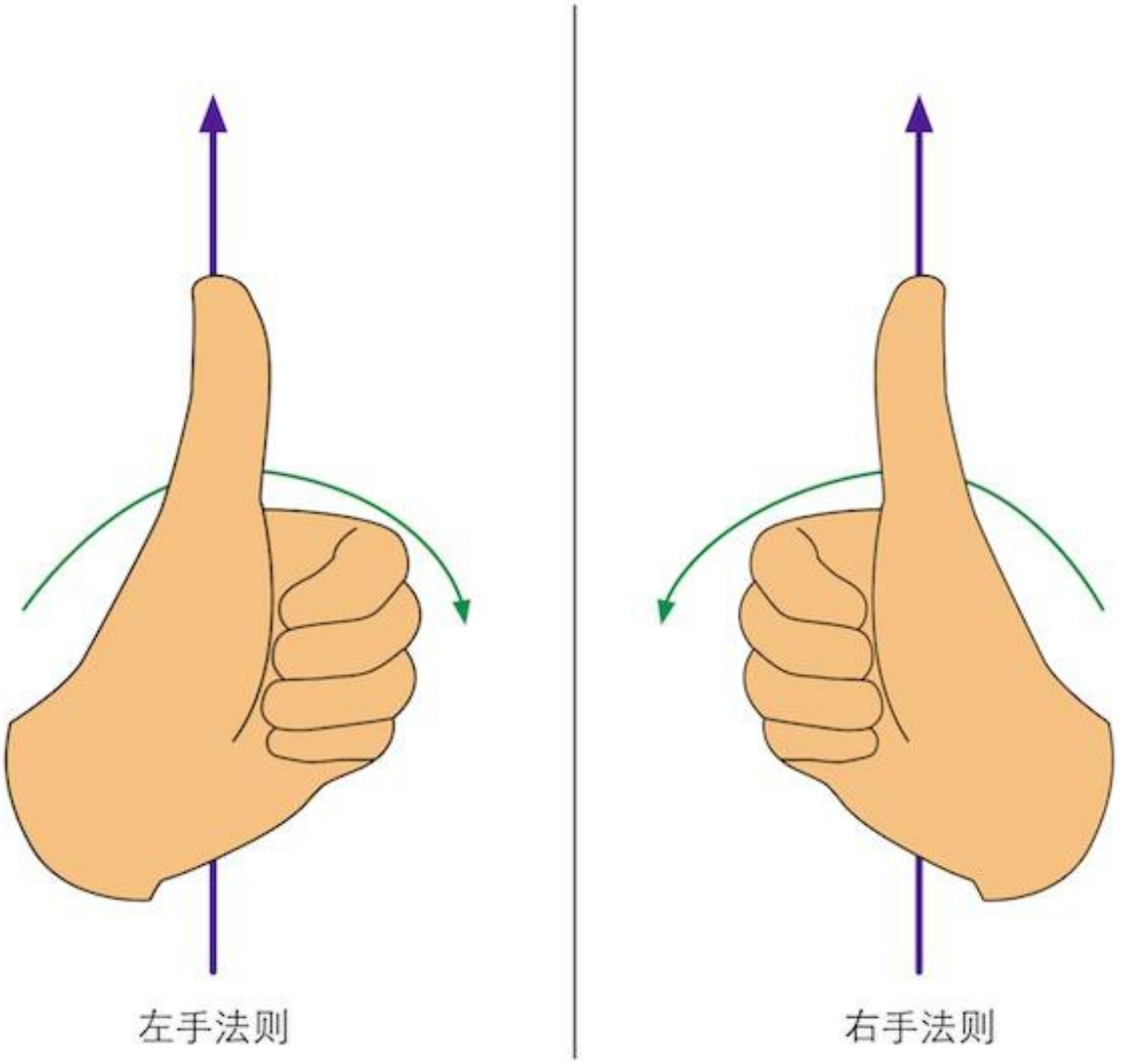
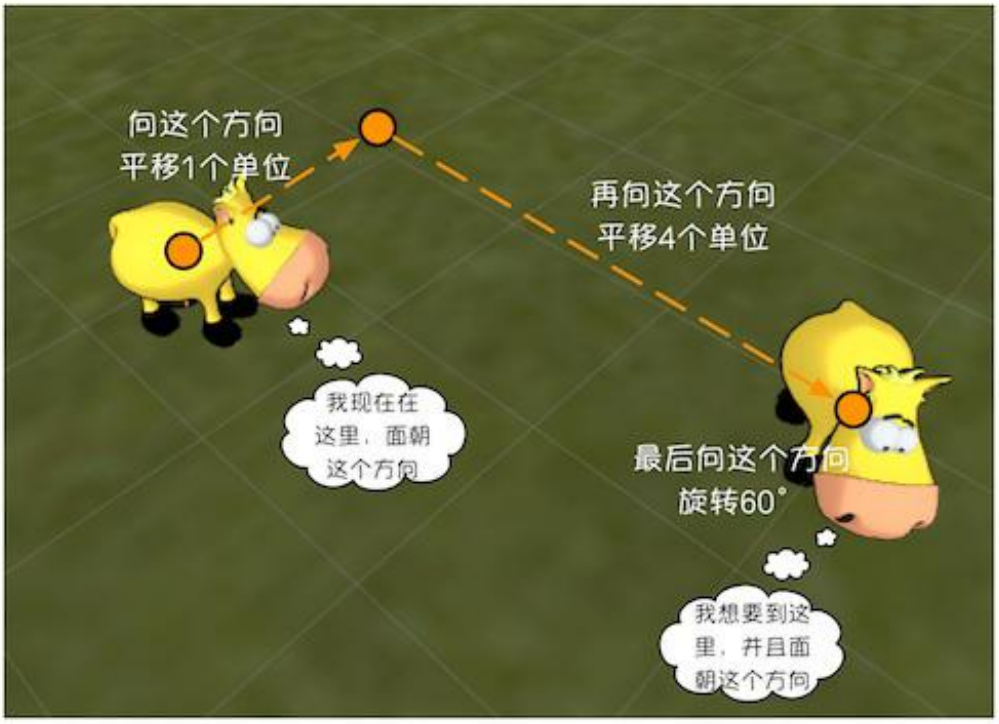


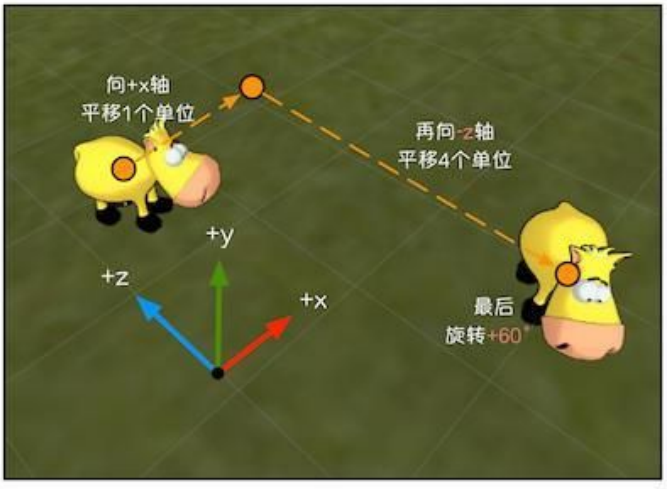
Figure 4.8right-handed

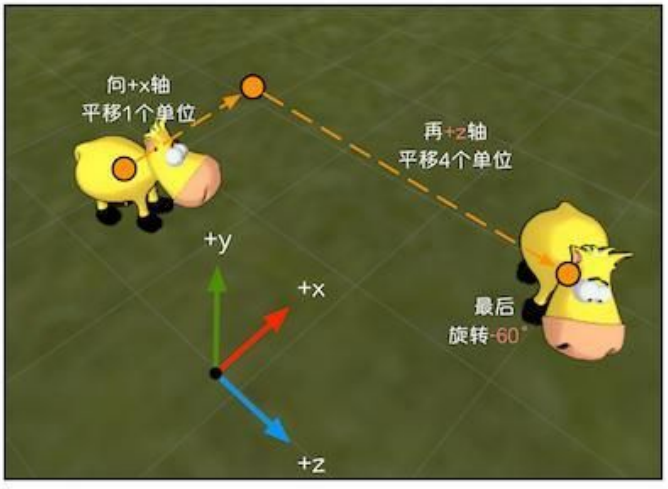


and left-hand rule with FIG. 4.9 right hand rule to determine the positive direction of rotation

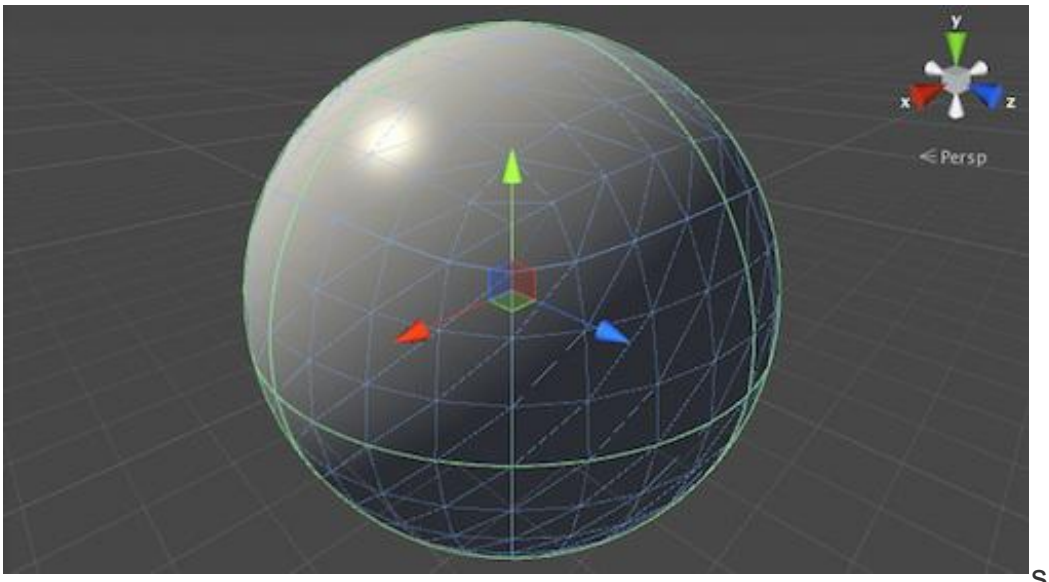
in order 

to move to the FIG. 4.10 the new location, niu units need to translate a certain direction, again the other direction translation unit 4, and finally again a rotation direction 60 °

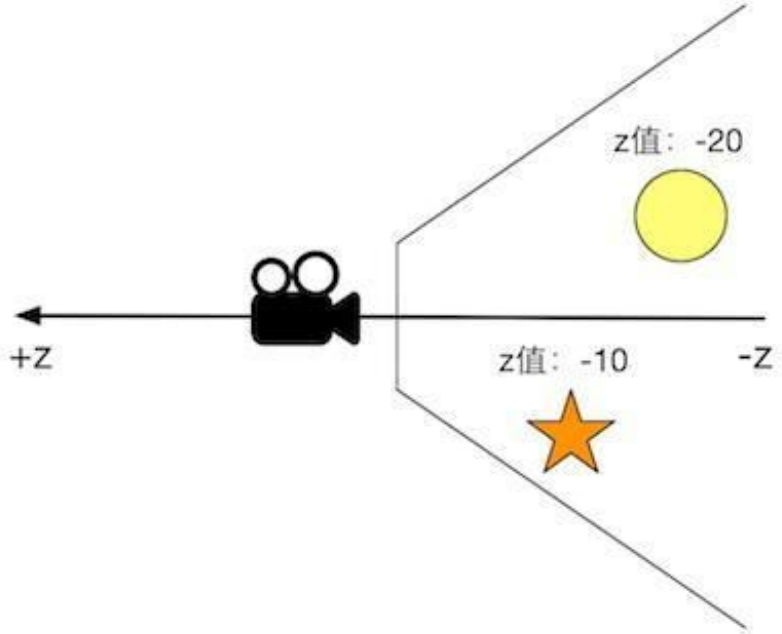




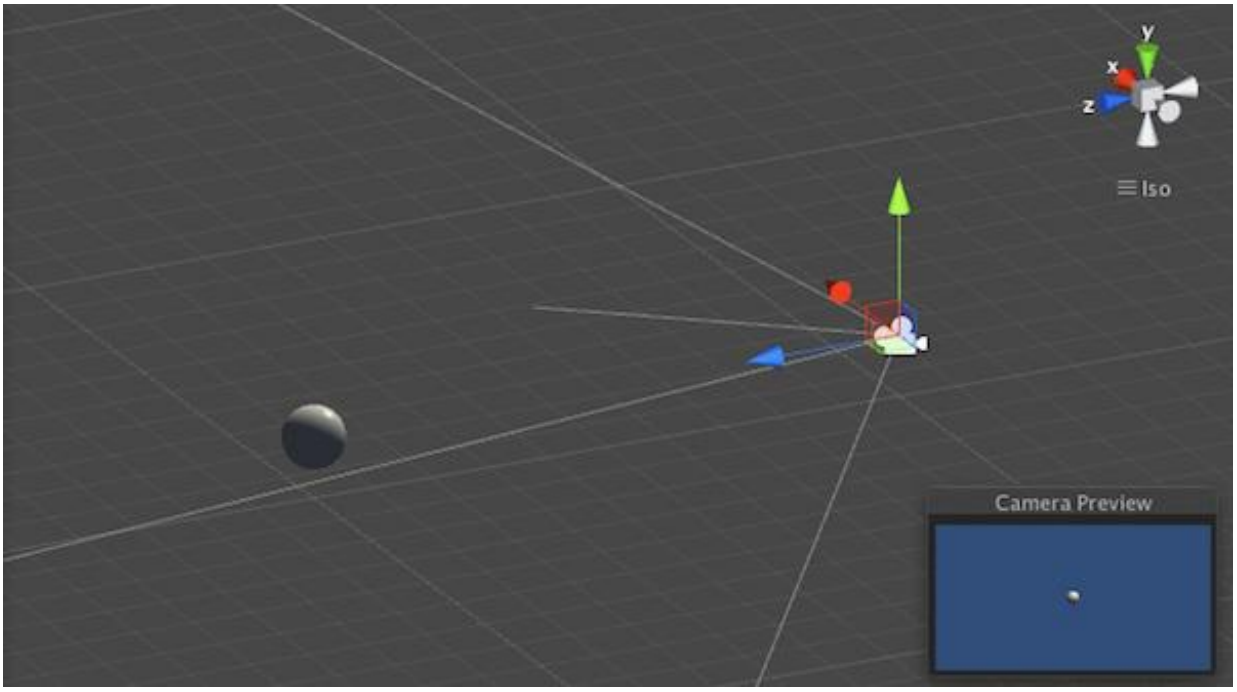
left and right, respectively, Figure 4.11 shows a left-handed and right hand coordinate system described niu result of this movement, the mathematical description to get a different

view of 

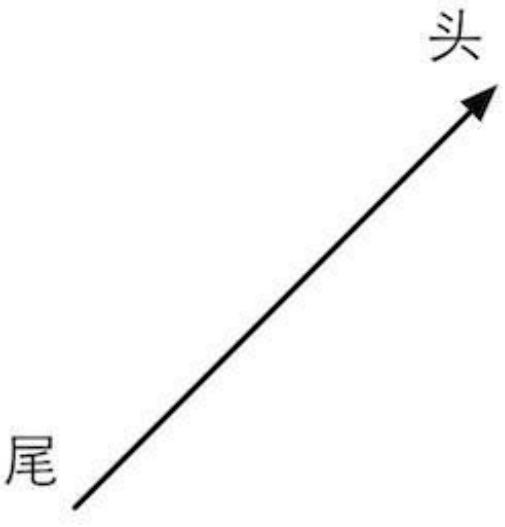
4.12 in model space and world space, Unity using a left-handed coordinate system. Drawing, the axis of the ball 3 which shows in model space coordinate axes (red x-axis, y-axis green, blue is the z-axis)



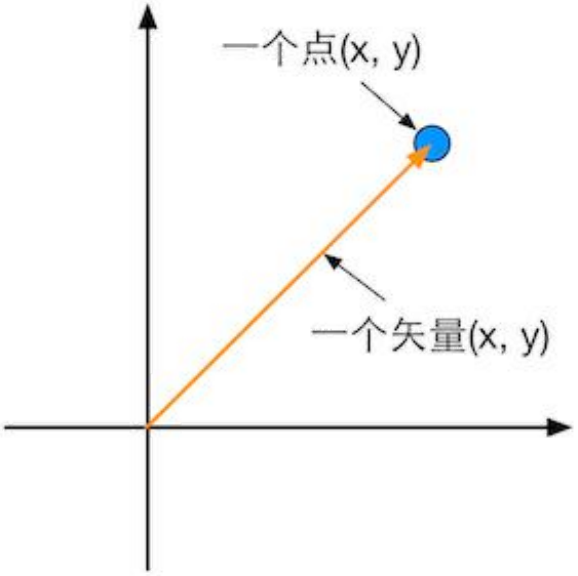
in Figure 4.13 Unity observed space using a right-handed , before the camera is the negative direction of the z-axis, z-axis, the larger the depth of the object, the farther away from thecamera



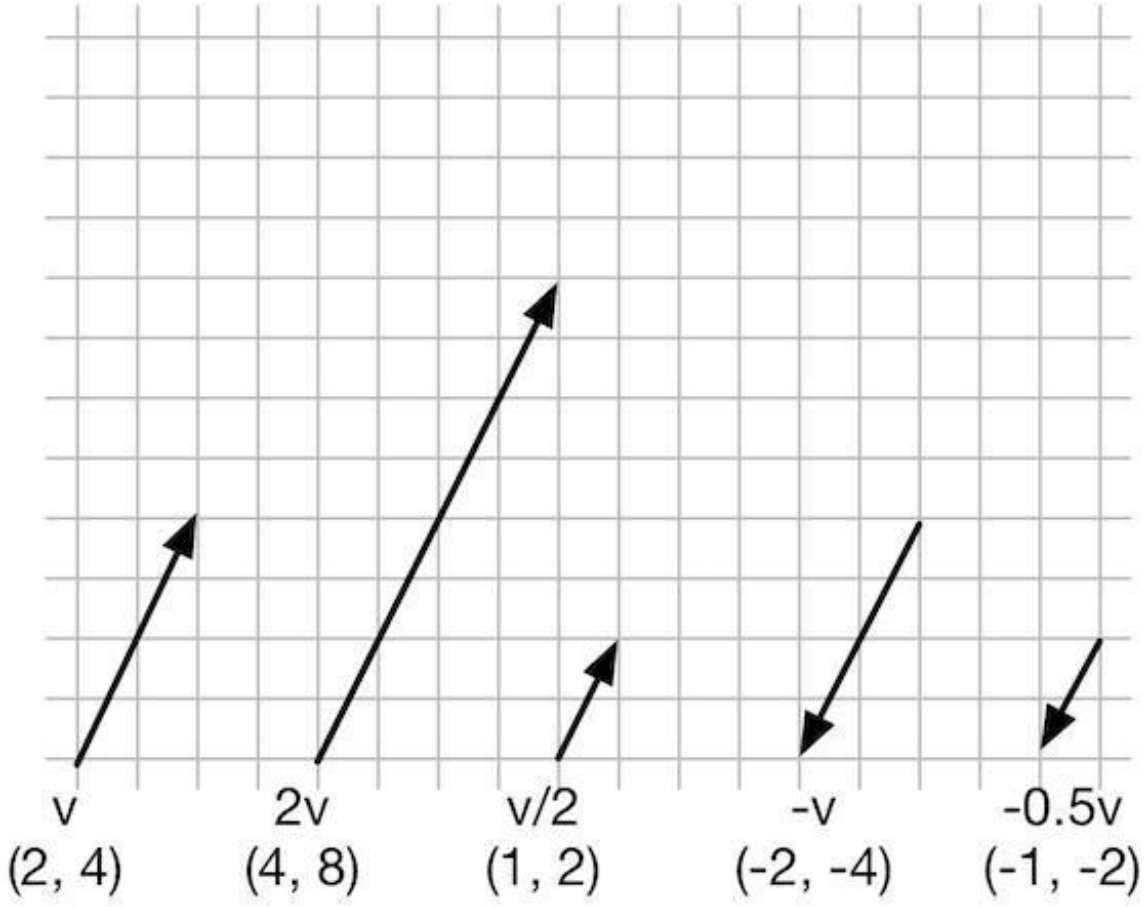
camera of FIG. 4.14position are (0, 1, -10), the position of the sphere is (0, 1 ,0)



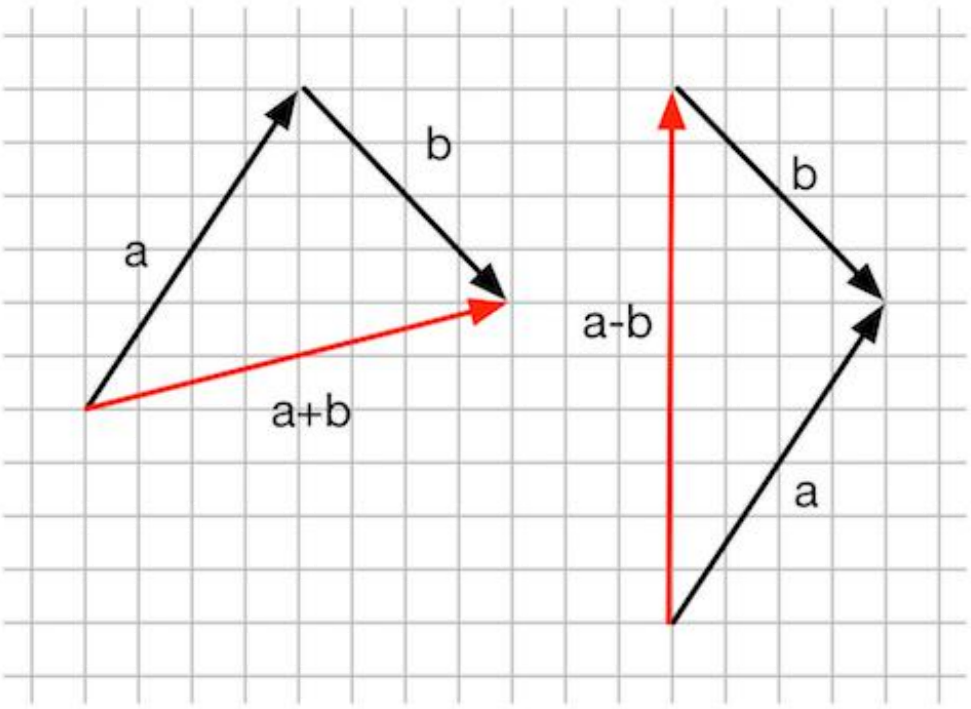
4.15 a two-dimensional vector and its head and tail



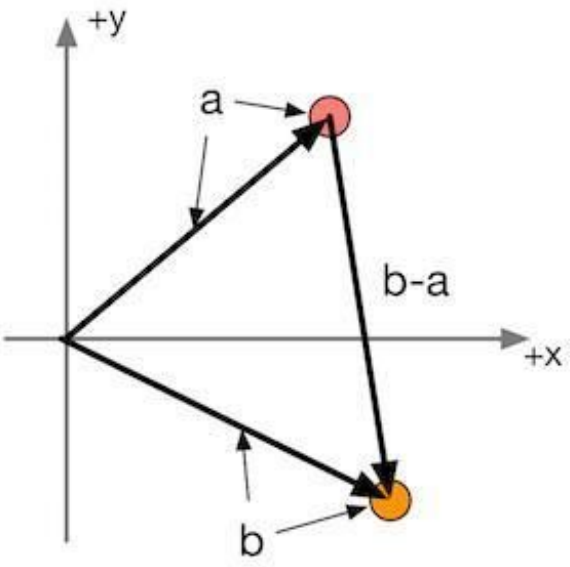
the relation thepoints and vectors FIGS 4.16



between4.17two-dimensional vector andnumber and scalar multiplicationdivision

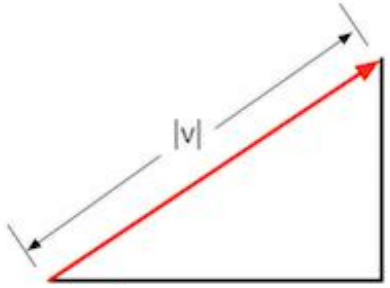


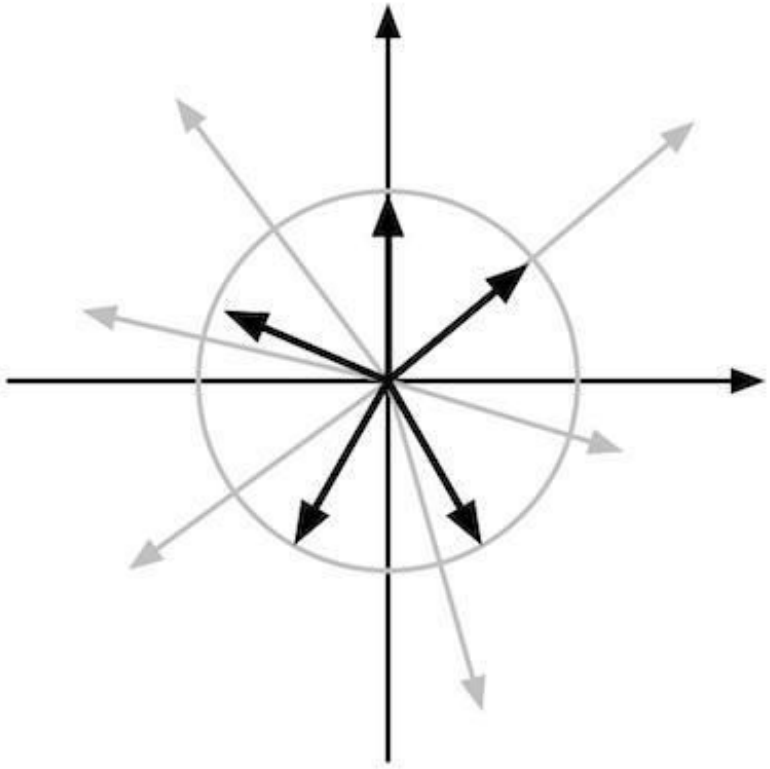
4.18aoftwo-dimensional vectoraddition and subtraction FIGS



using vector subtraction 4.19 is calculated from the point a to point bdisplacement

diemap4.20vector





in FIGunits 4.21dimensionalspace vectorwill fall on the unit circle

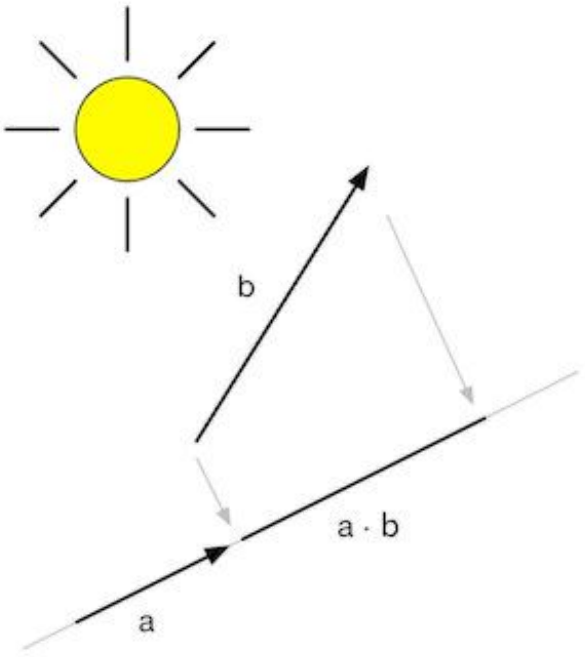
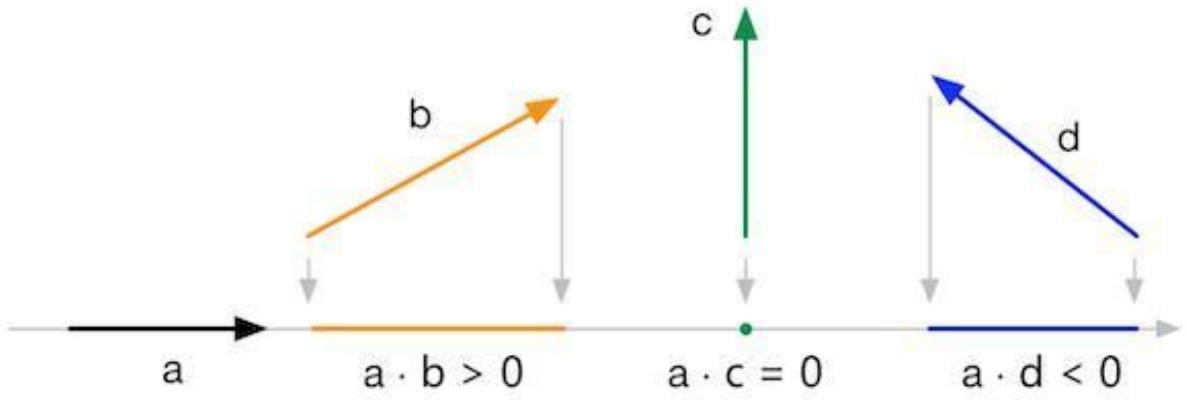
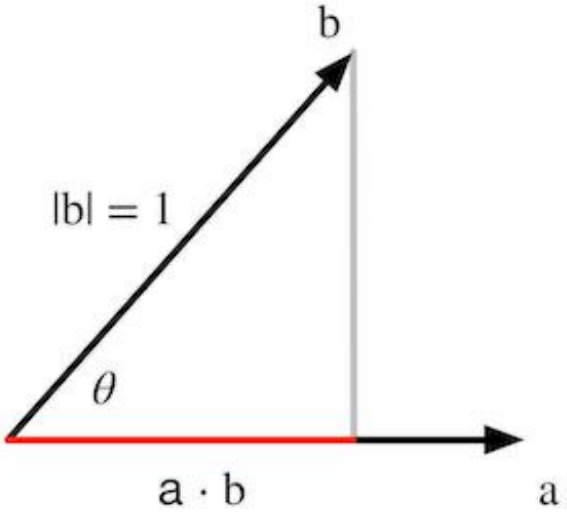


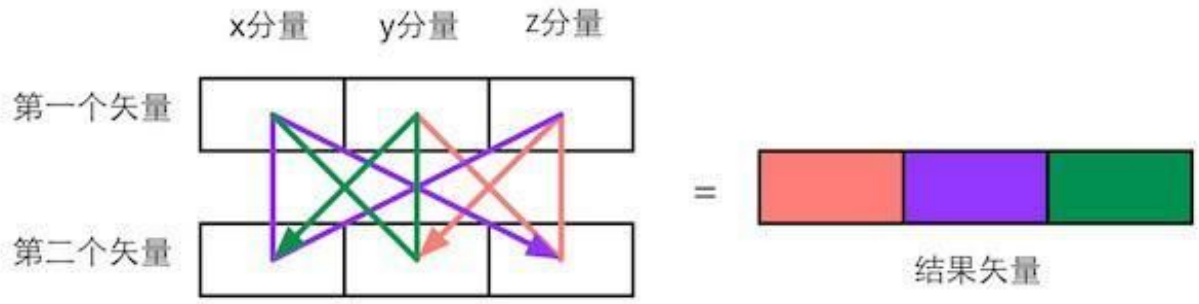
FIG vector b 4.22 a unit vector in a direction ofof



notationprojectionthe dot product4.23

in FIG.two 4.24 dot 

product of unit vectors



4.25 FIG three-dimensional vector cross product calculation rule. Different colored lines represent the calculation result of computation of a path component vector corresponding color. Red, for example, the first component i.e. the result vector, which is multiplied by the z-component of the starting components of the second vector from a vector y, z components minus the first vector and the second vector is the product ofy component of

FIG4.26 using the vector a and vector b construct a parallelogram

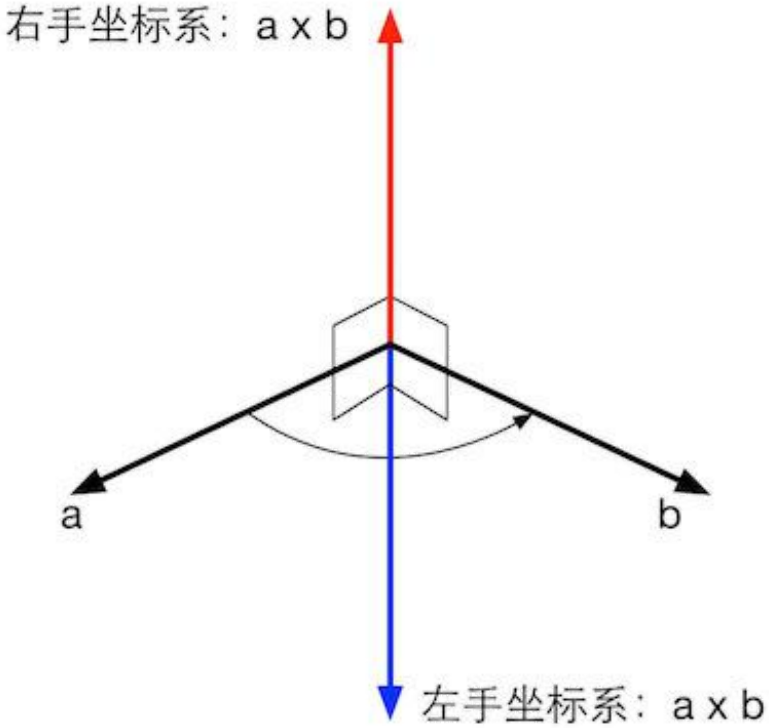
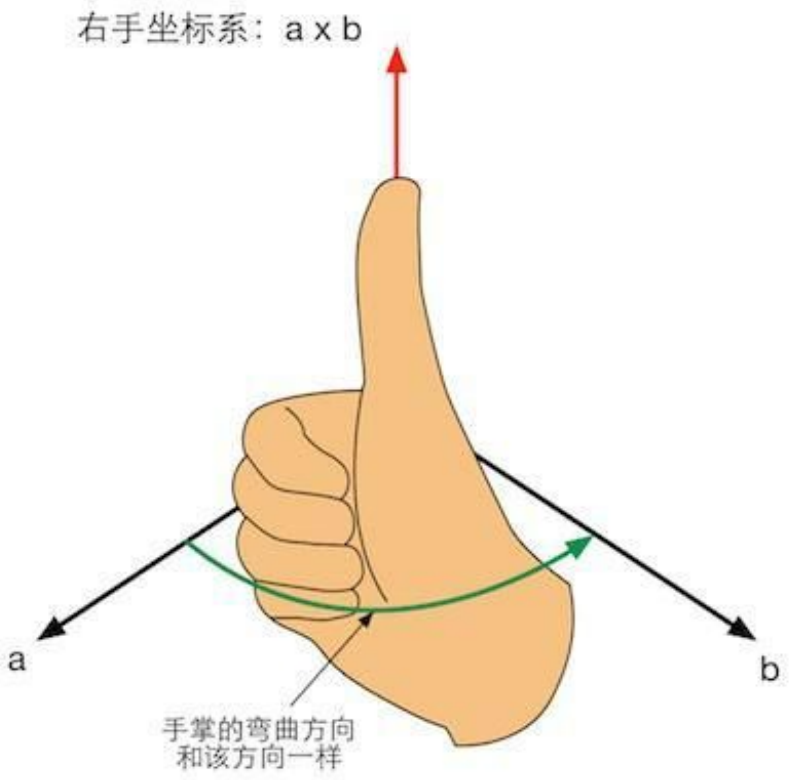
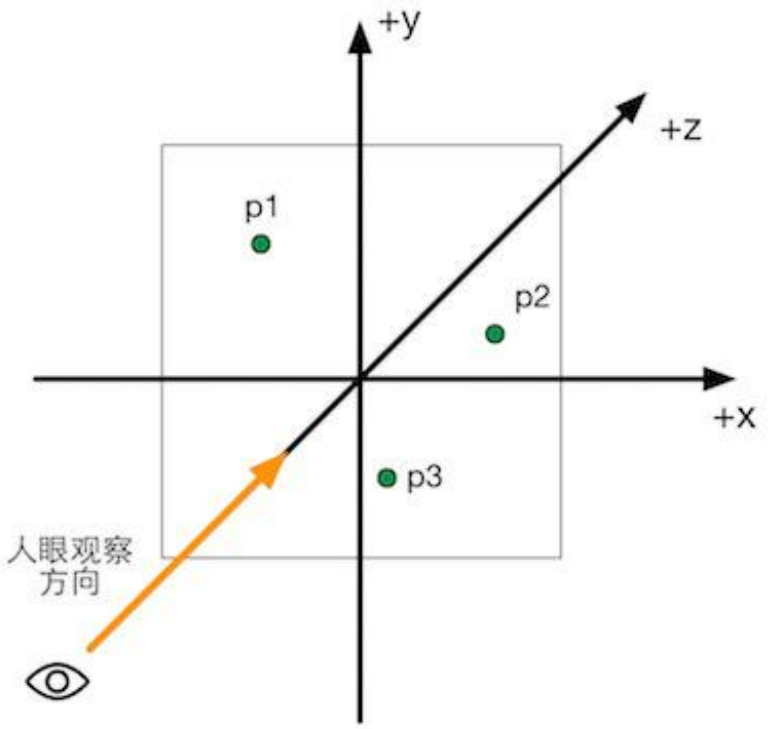


FIG. 4.27, respectively, using the cross product resultsleft-handed and right-handed coordinate system obtained



in FIG4.28 using the right hand rule determines right-handed coordinate system a × b direction

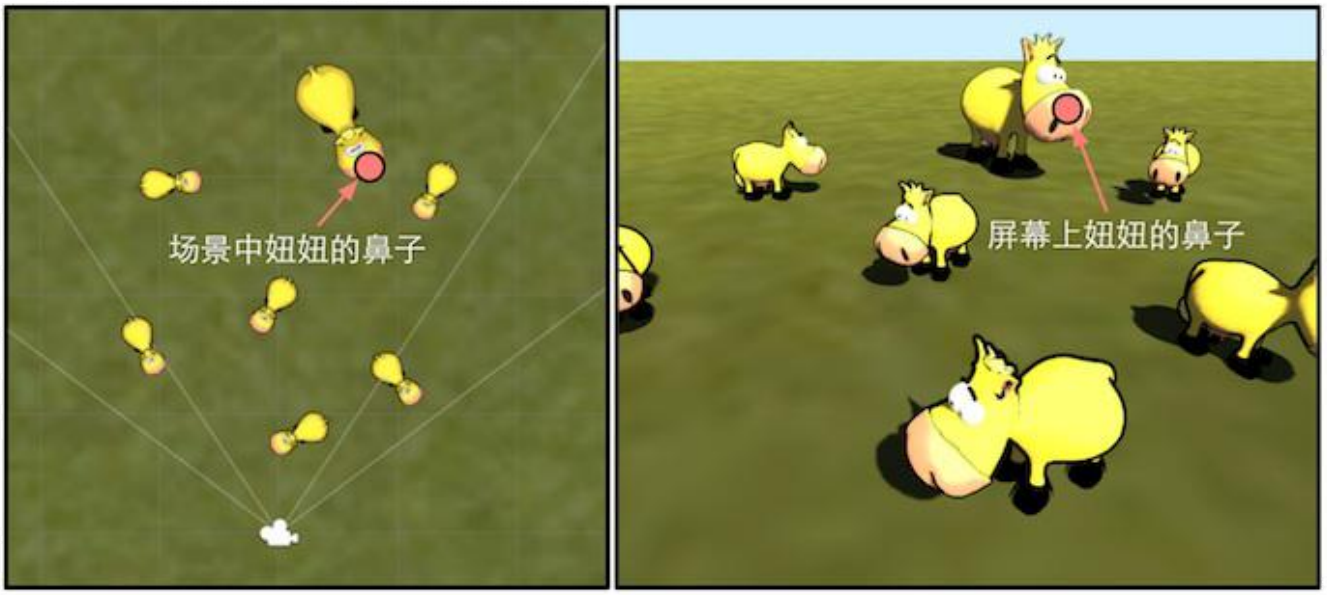


in FIG. 4.29 triangle three vertices located on the xy plane, eye z-axis negative direction is located, viewedin the positive direction of the

$$\begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} \quad \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

in FIG calculation c23 4.30process



4.31 scenein FIG niu (left) and niu (right) on the screenz-axis.Niu wondered how his nose is drawn to theon the screen of

map4.32 in our farm game, each cow has its own model coordinate system. Niu in the model coordinate system is the nose position (0, 2, 4, 1)

in

FIG Transform 4.33 Unity assembly may adjust the position of the model. If there is a parent node Transform, as shown in the "Mesh", it will be in Position model space parent node (here "Cow") in the position; if no parent node, position is the position in world space

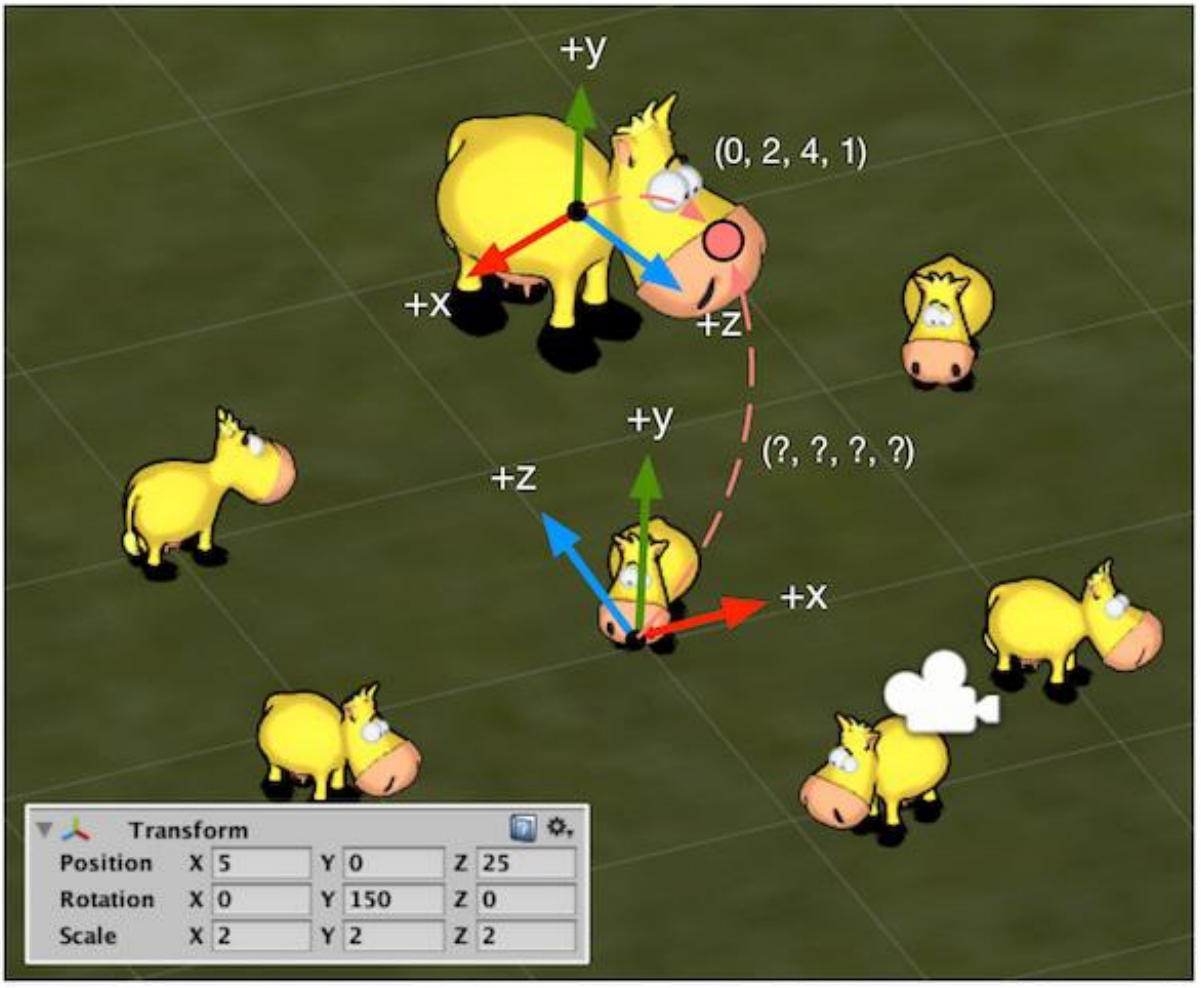
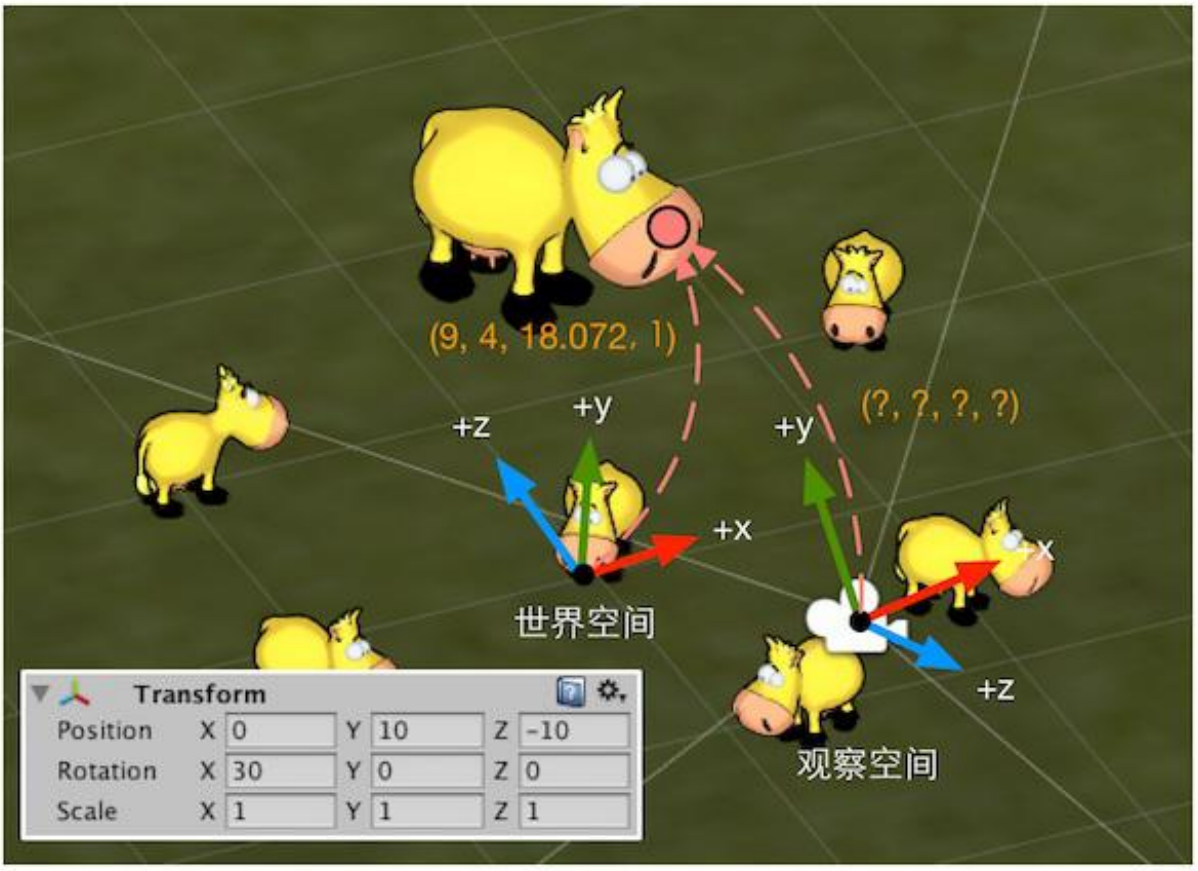
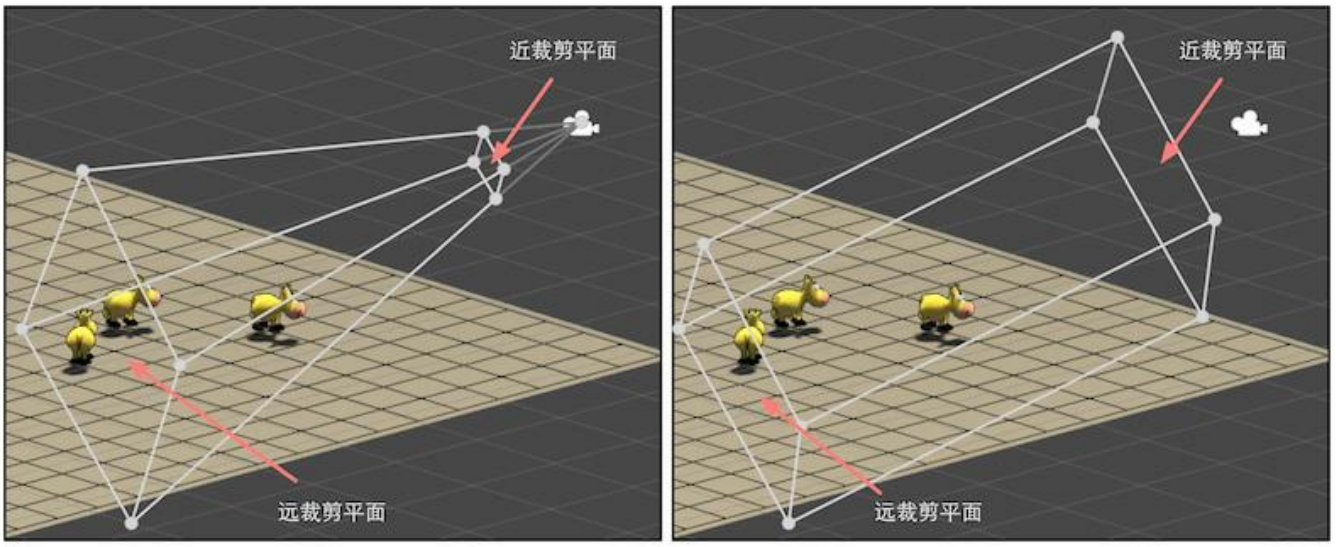


Figure 4.34 farm game world space. The origin of world space is placed in the center of the farm. The lower left corner shows the transformation made niu in world space. We want to niu nose transformation from model space to world space

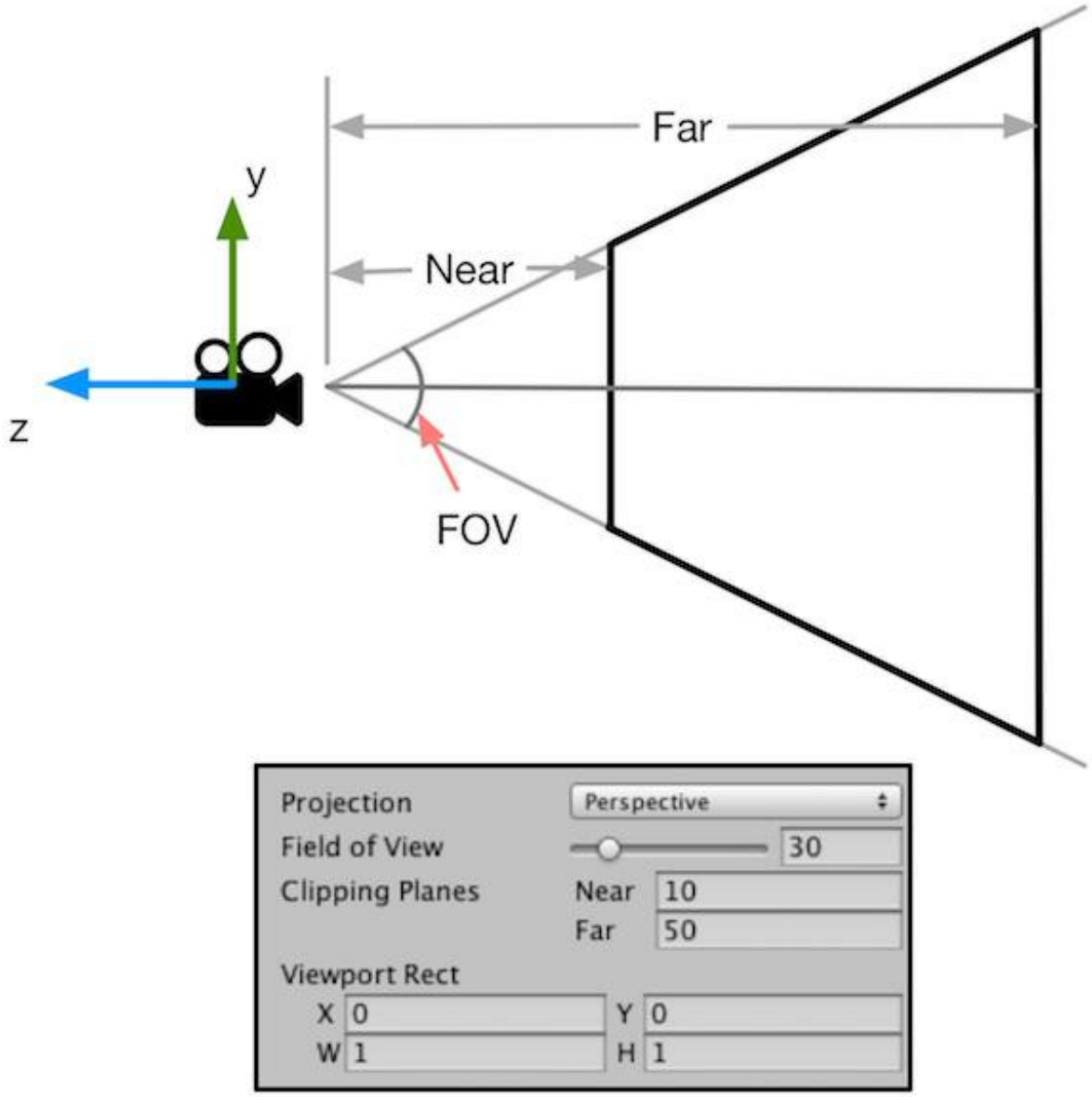


observation space Figure 4.35 game farm in the camera. Origin of the observed space located at the camera. Note that in viewing space, forward of the camera is the negative direction of the z-axis (shown only in FIG z-axis positive direction), because the use of right-handed Unity in viewing space. The lower left corner shows the transformation made by a camera in world space. We want to niu nose transformation from world space to viewing space

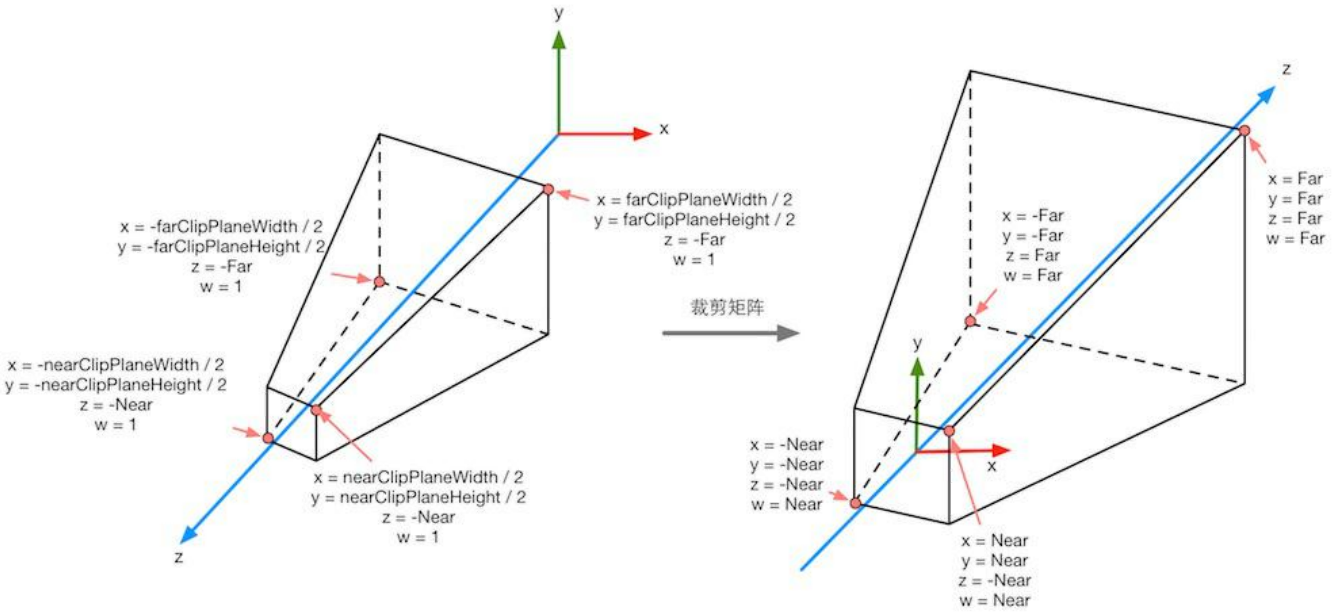
perspective projection Figure 4.36 (left) and orthogonal projection (right). The lower left corner show the projection of the current mode of the camera and associated attributes



and FIG. 4.37 frustum clipping plane. The left panel shows a perspective view of the cone projection, right panel showsorthogonal projectionview frustum

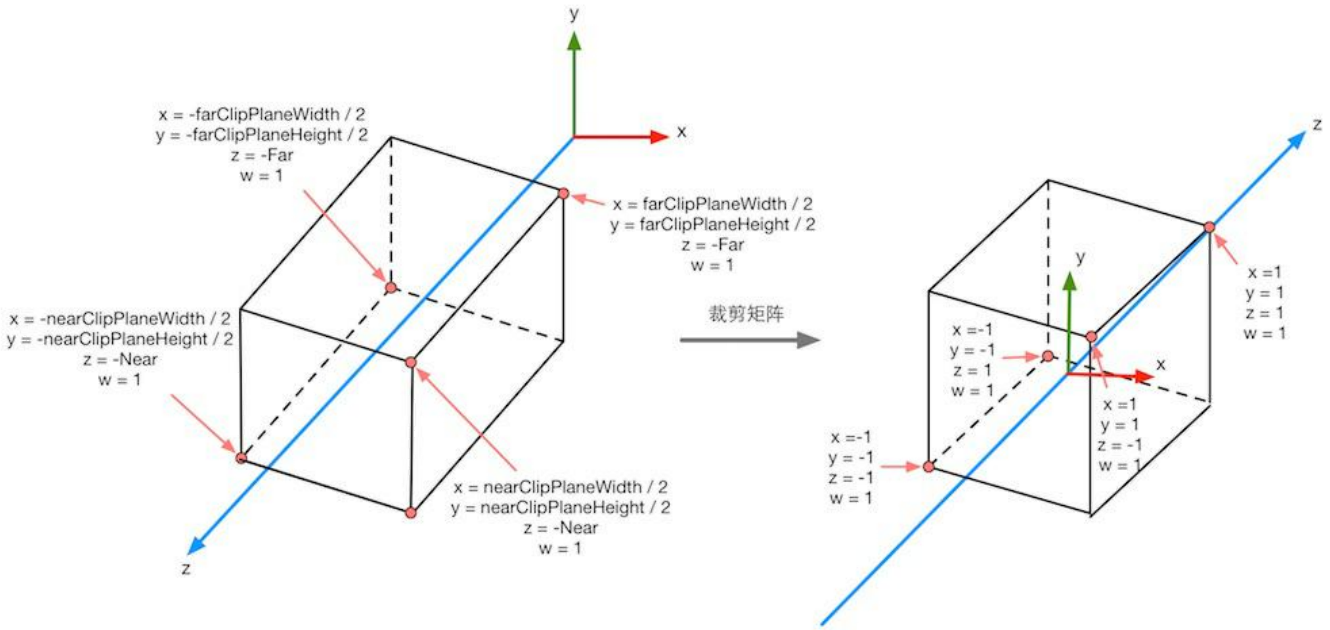


a perspective view of the camera parameters 4.38of FIG perspectivecone impact cast film



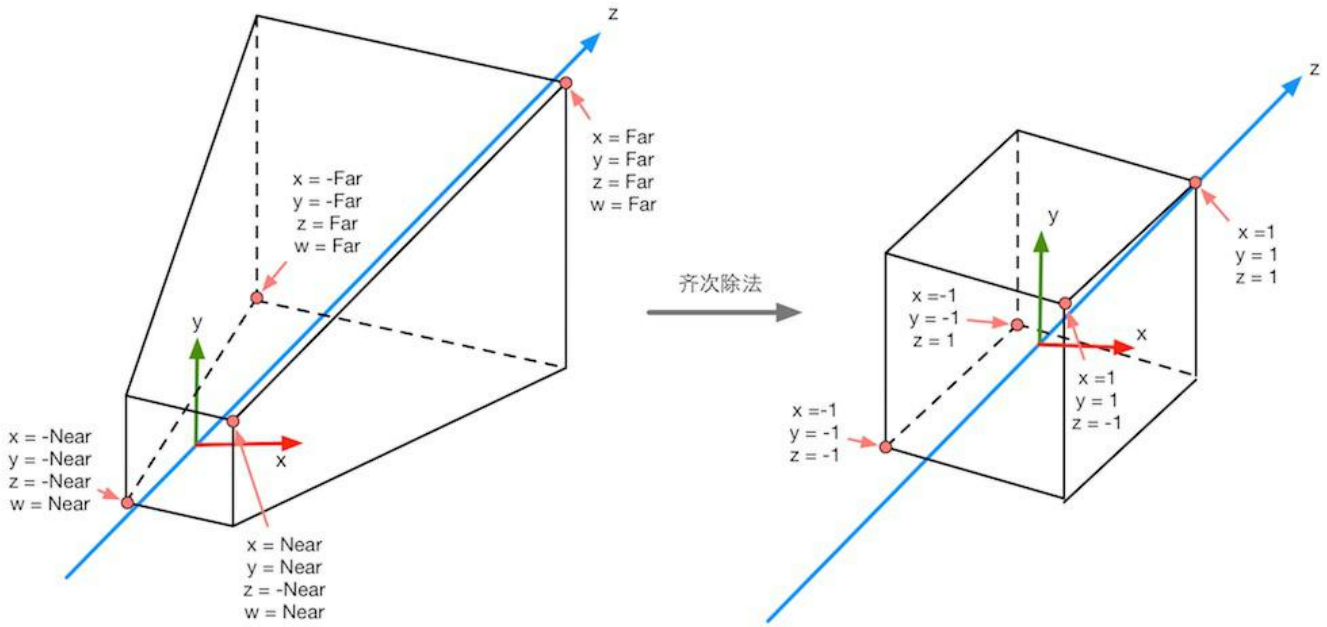
in FIG. 4.39 perspective projection, the projection matrix is scaled vertices . 3.38 noted in FIG four key points after the result of the projection matrix transformation. From these results it can be seenvariationrange ofx, y, z, and w components occurring

the camera parameters 4.40 orthogonal FIGon the positive influence tradingfilm cone



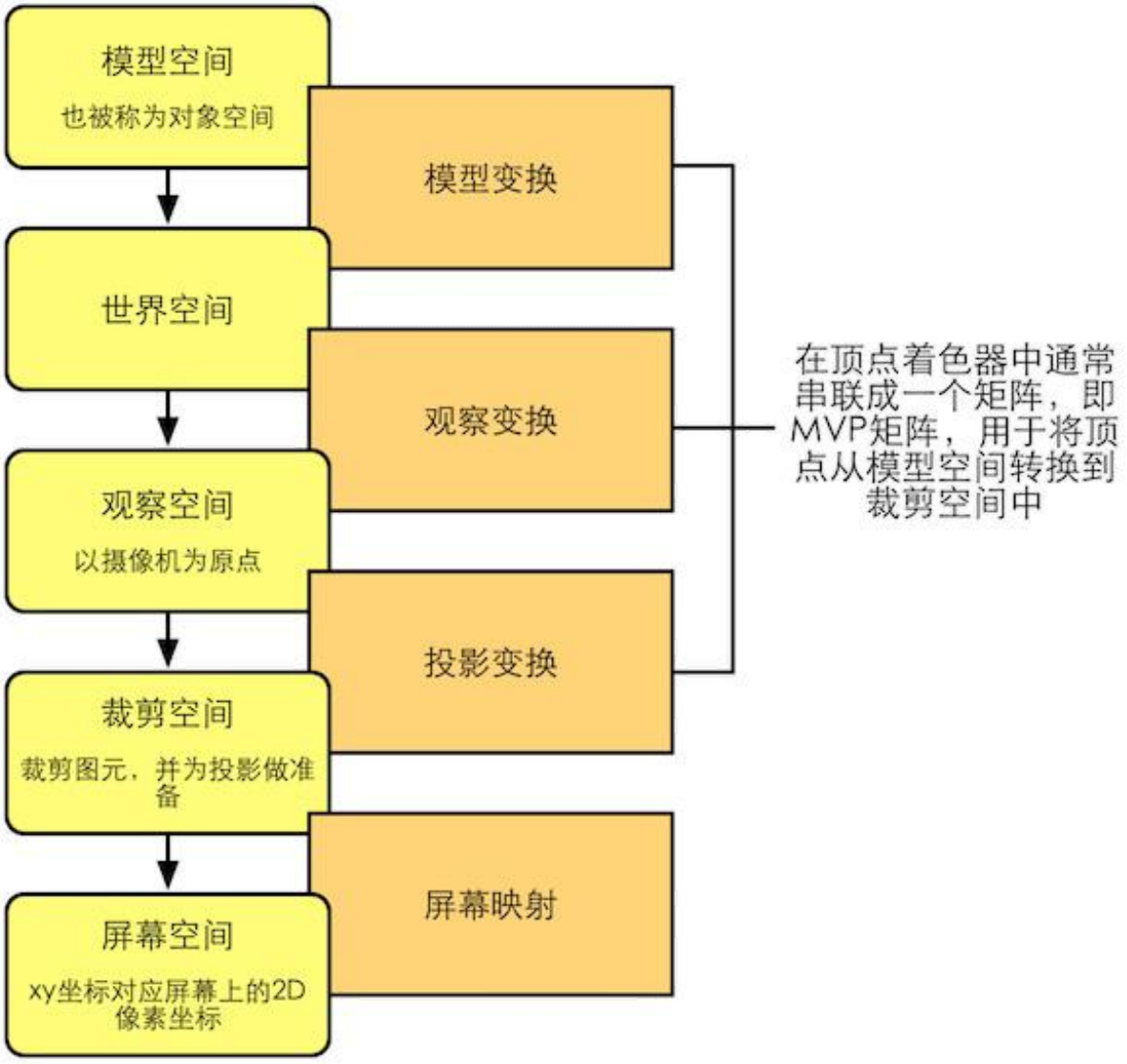
in FIG4.41 orthogonal projection, the projection matrix is scaled vertices . FIG noted in the results after four key points after projection transformation matrix. These results can be seen fromchangesx, y, z and w component of the range ofoccurrence

the aspect of the camera parameters and the game screenFIG4.42 farm gameratio used

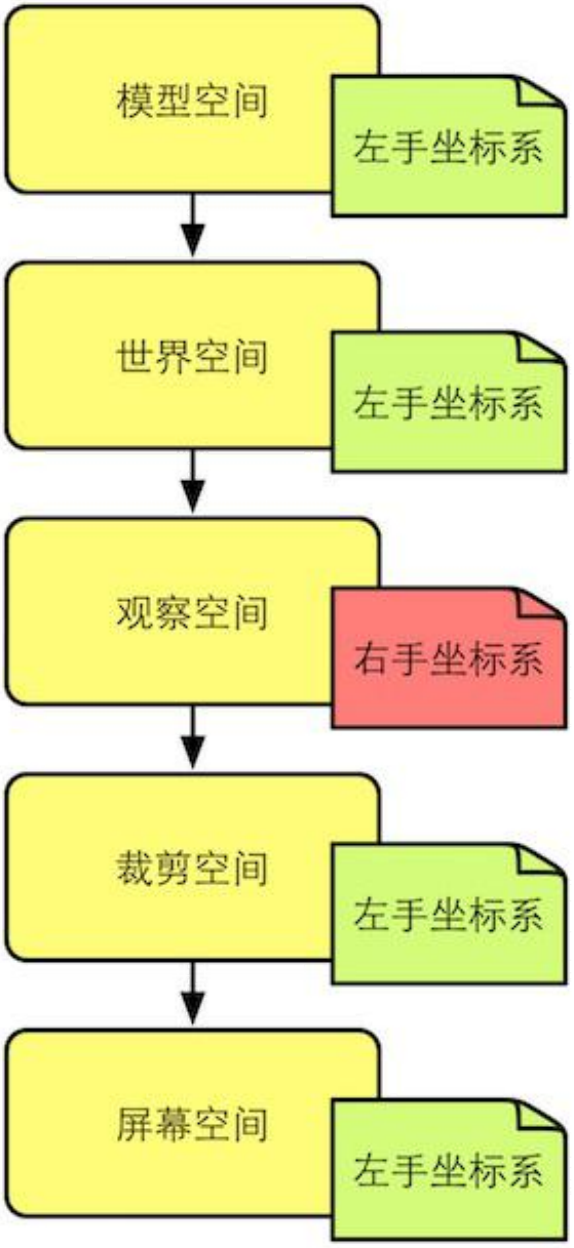


in FIG division homogeneous after 4.43, perspective projection conversion will be cut to a spacecube

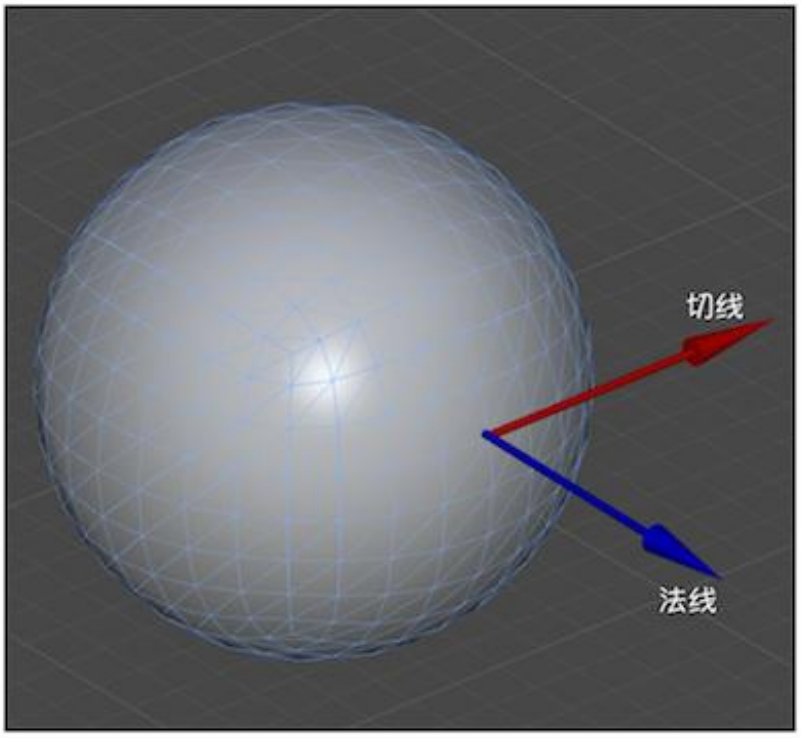
FIGdivision homogeneous after 4.44, clip space will be transformed into orthogonal projection ofacubic



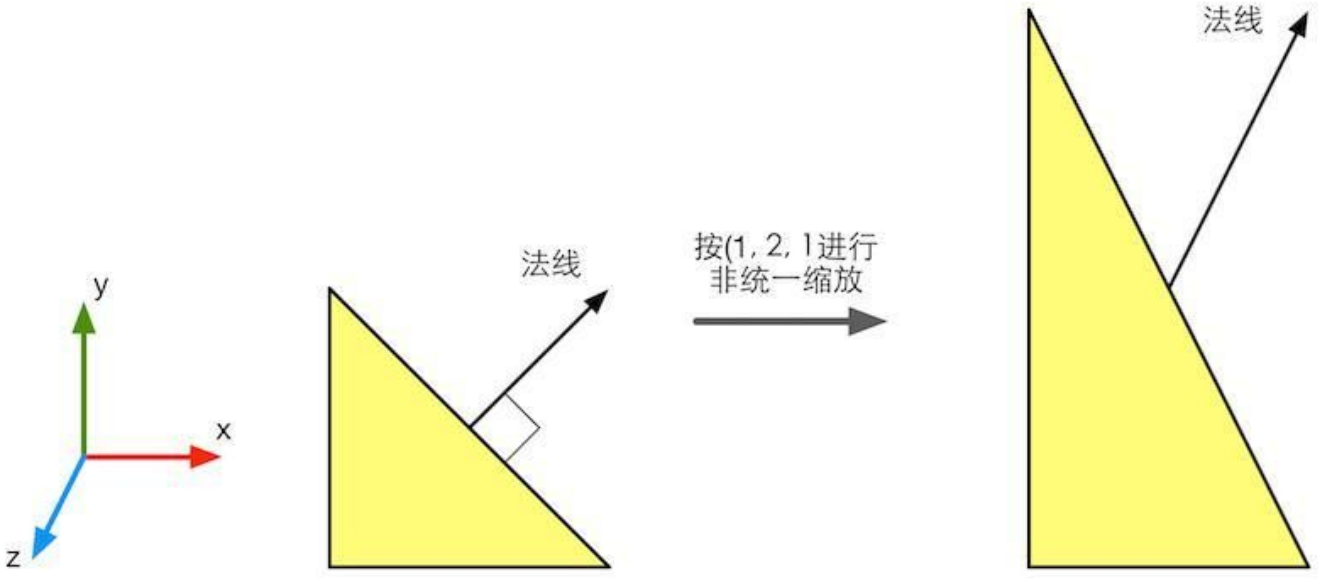
of the rendering pipeline4.45FIG spatial transformation process vertices



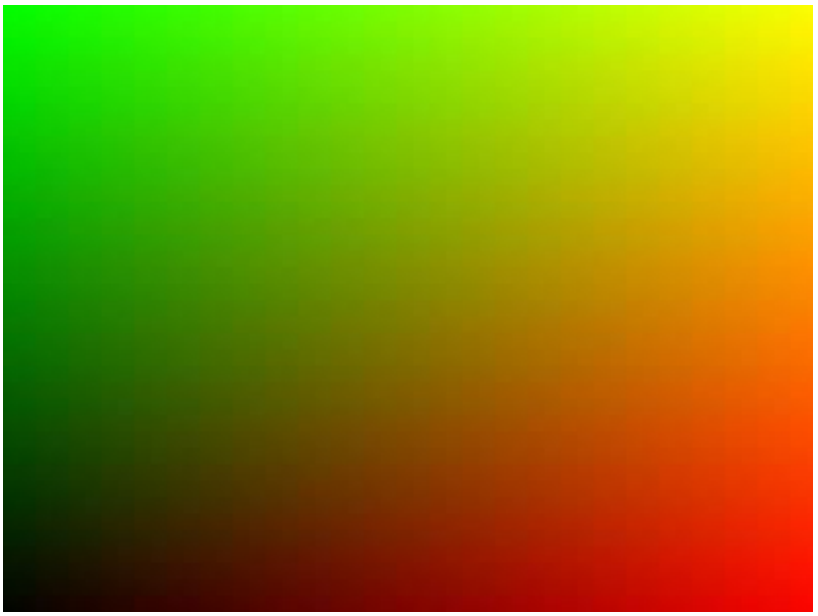
handedness each coordinate spacein4.46 Unity FIG

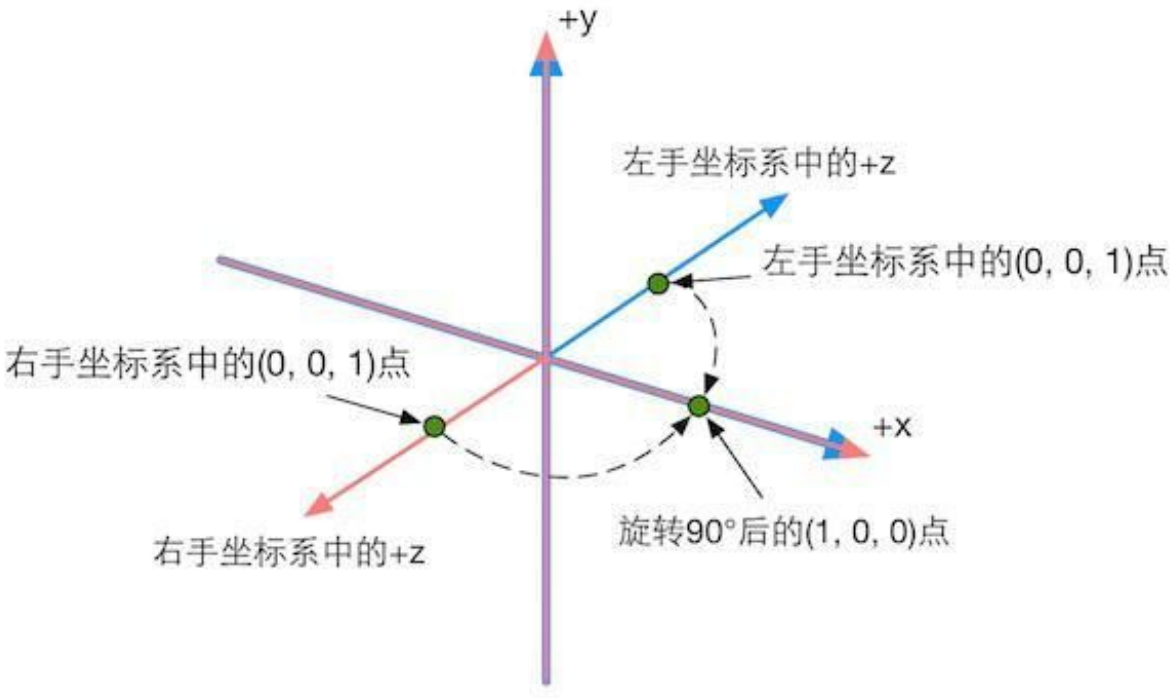
FIG4.47vertex 

tangent and normal.each tangent and normal vertical

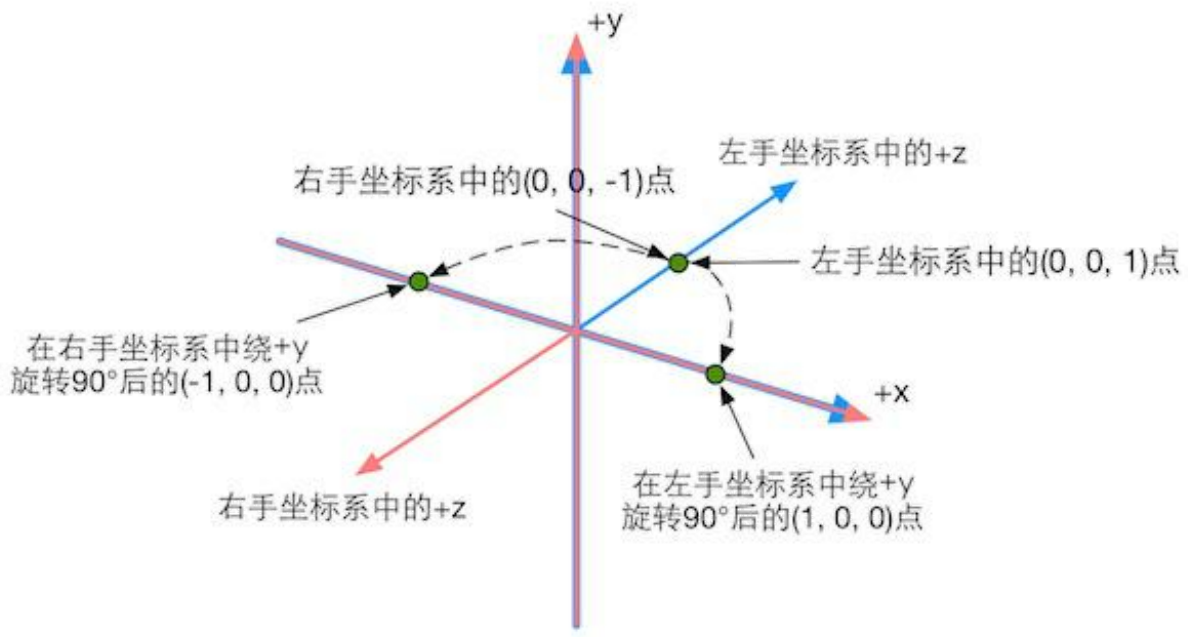


Whennon-uniform scaling FIG. 4.48, and if using the same transformation matrix transforms transformed vertex normals, will give erroneous results, i.e., the direction normal to the plane is no longer perpendicular toconverted

FIGby a 4.49tile  image pixel position obtained

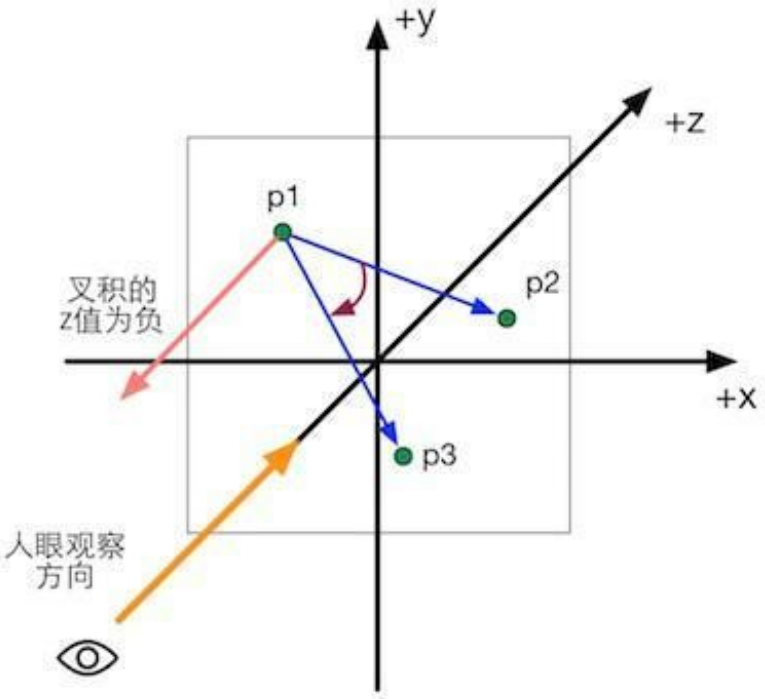


FIGin FIG. 4.50 x and y axes of the two coordinate systems coincide, except that the direction of the z-axis only. Left-handed coordinate system (0, 0, 1) point and the right-handed coordinate system (0, 0, 1) points are different, but the point after they are rotated, but correspond to the same point



in

FIG. 4.51 at the same point in an absolute space, the same rotation angle in the left and right hand coordinate system, the direction of rotation is not the same. Clockwise rotation will be in the left-hand coordinate system, the right-handed coordinate system rotate counter-clockwise



in FIG left-handed 4.52, if the result of the cross product is negative, then the order of 3:00 clockwise

Chapter 5 start Unity tripshader learning

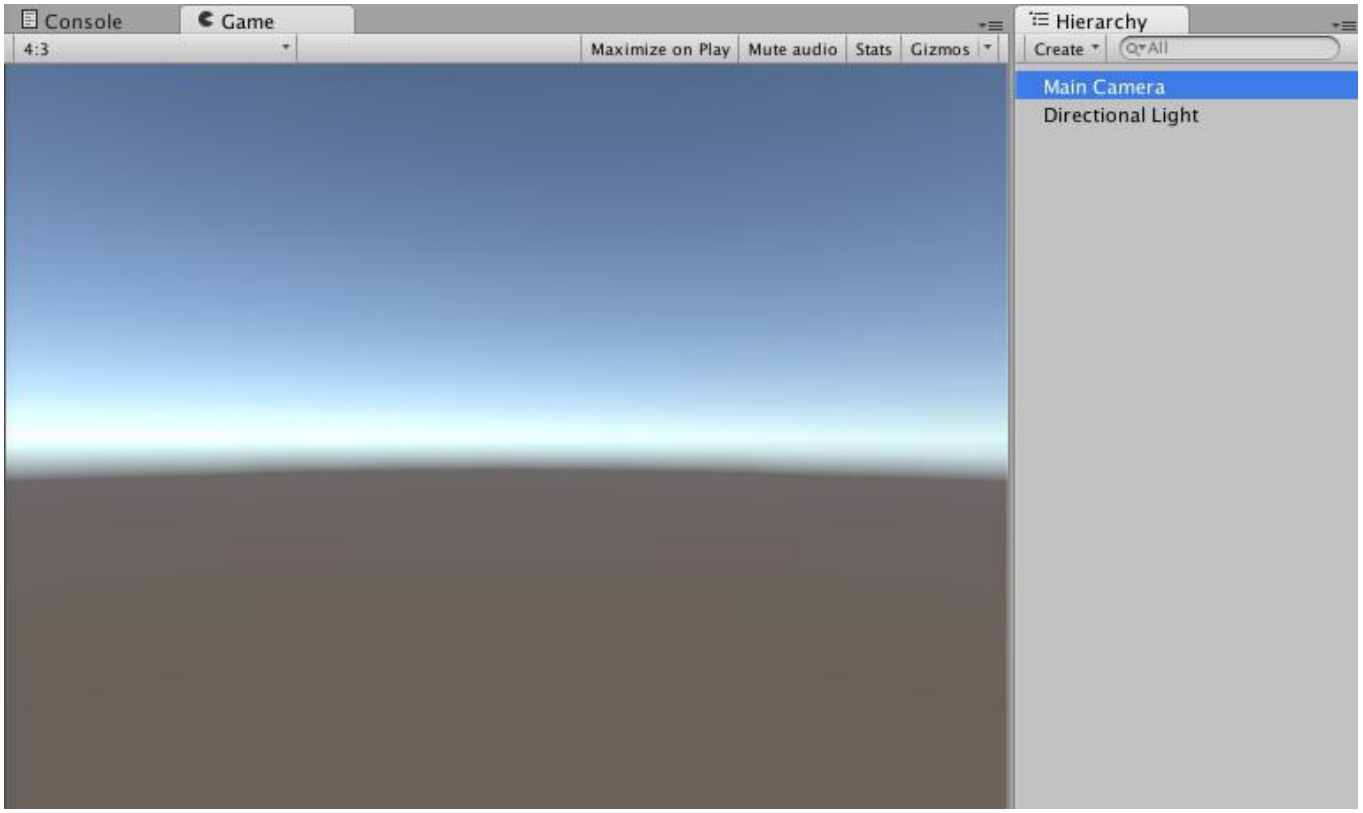
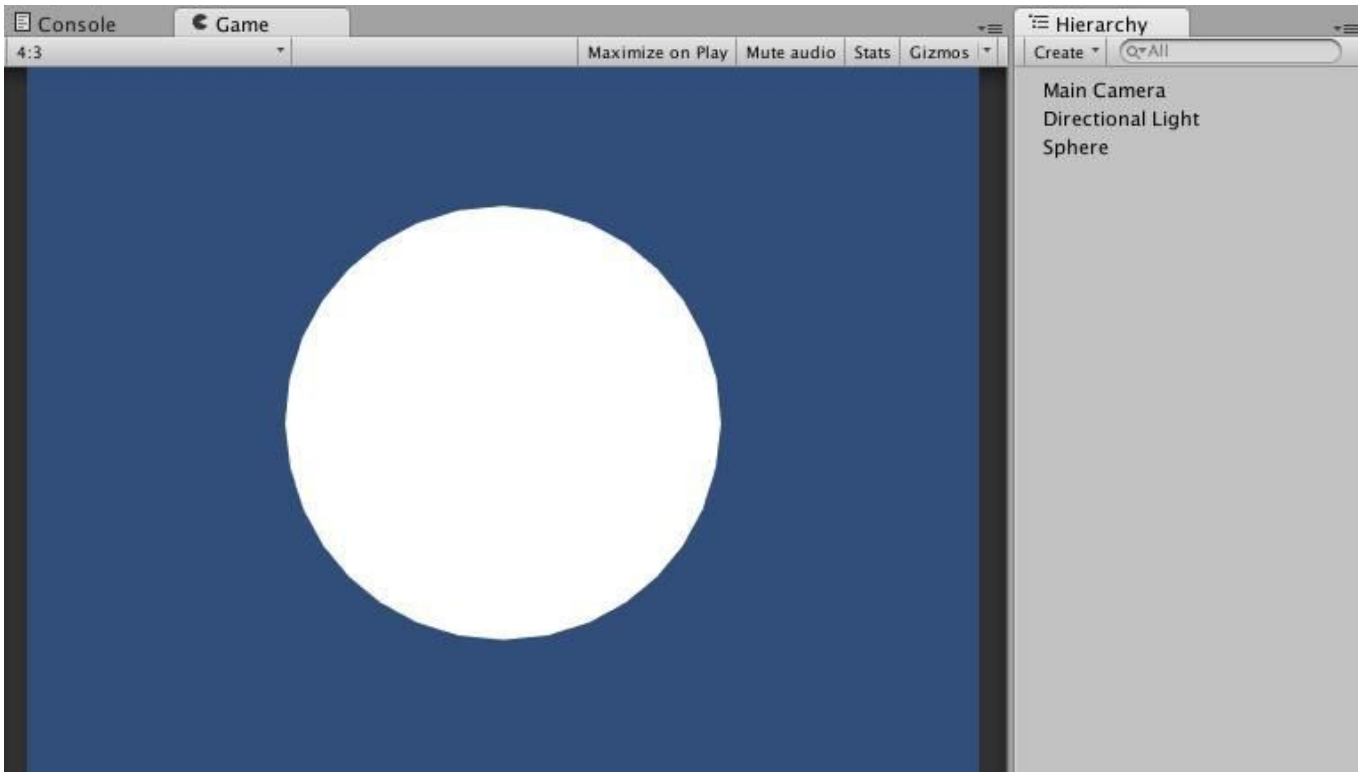
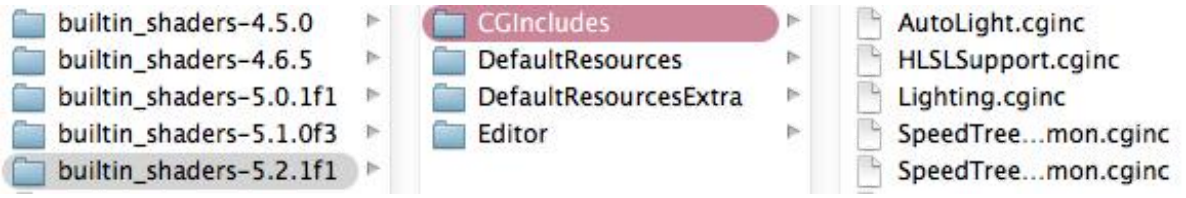


FIG5.1 new a scene obtainedvertexUnity. 5 in



rendering5.2 using a simple / fragment shader to give a white ball

FIG 5.3



Unity built shader

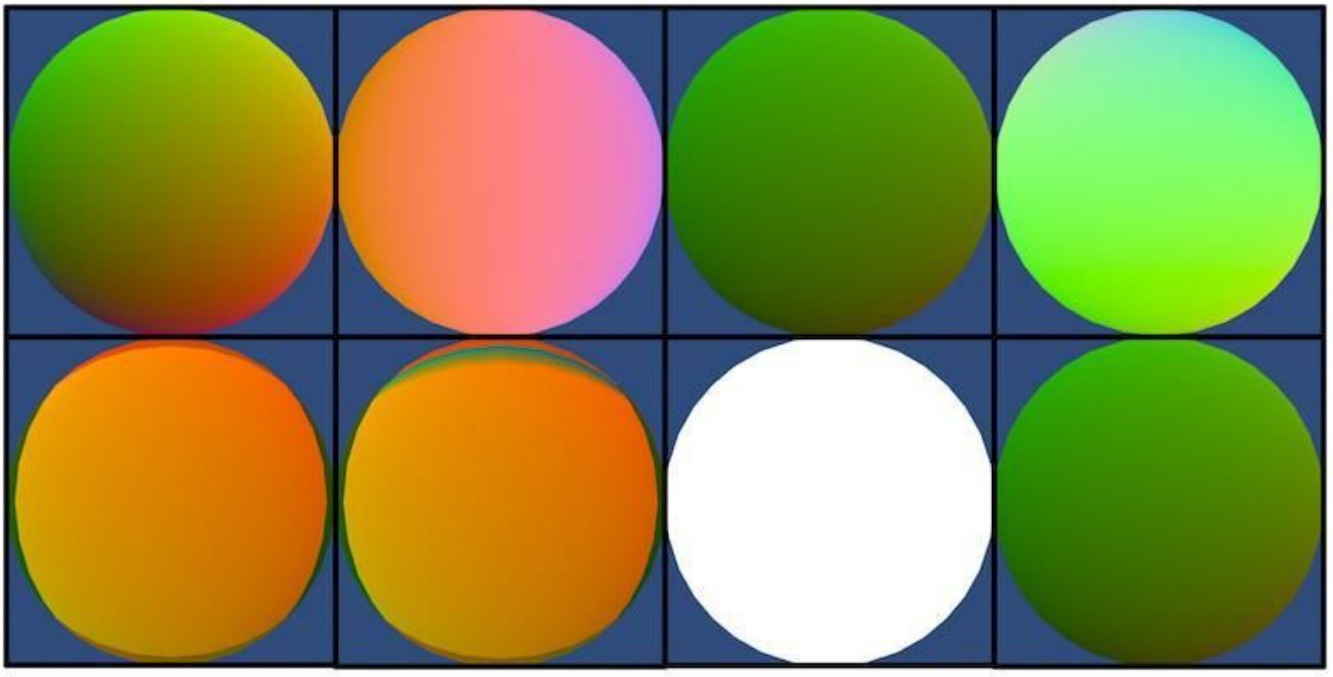
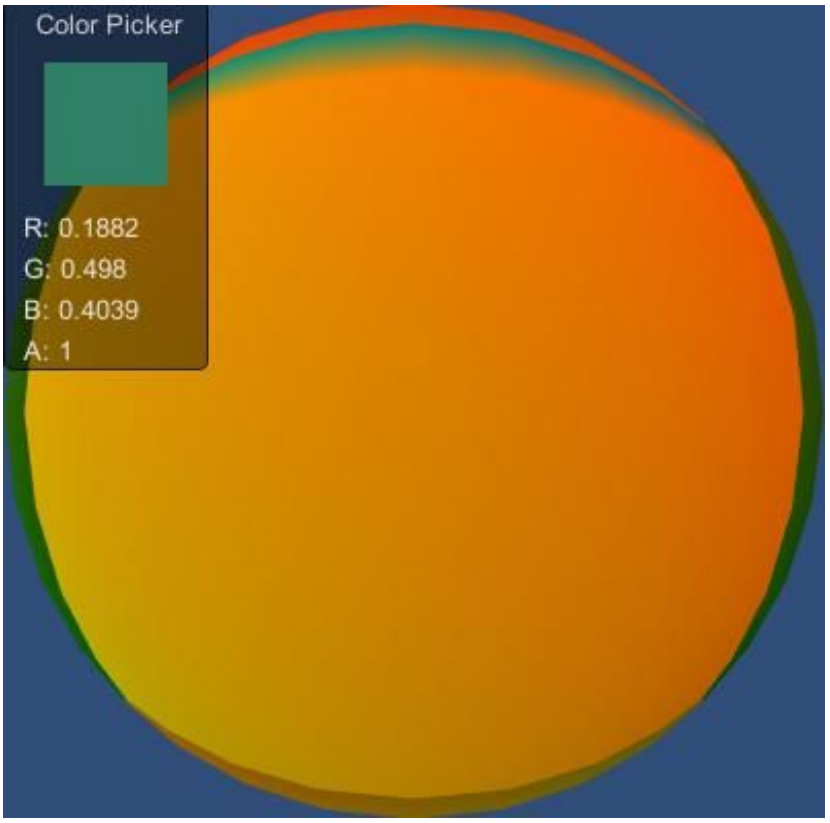


FIG 5.4 with false color of UnityShader debugging

Figure 5.5using 

the color picker to view debugging information

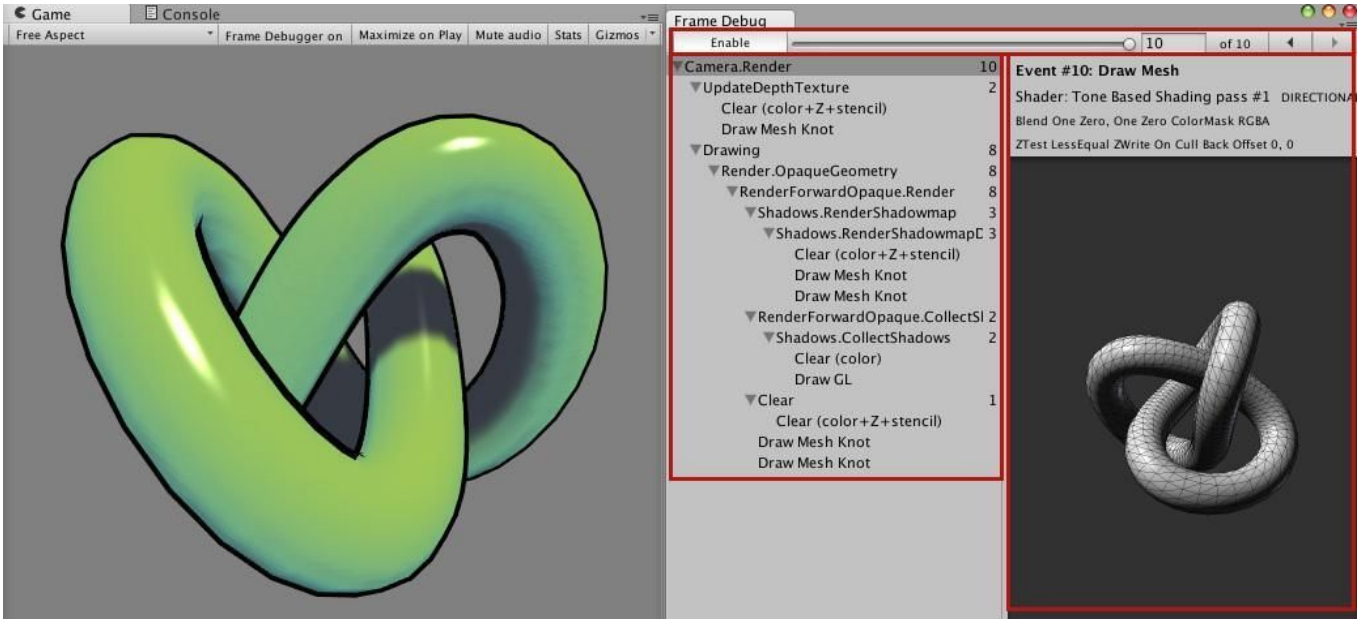


Figure 5.6 debugger

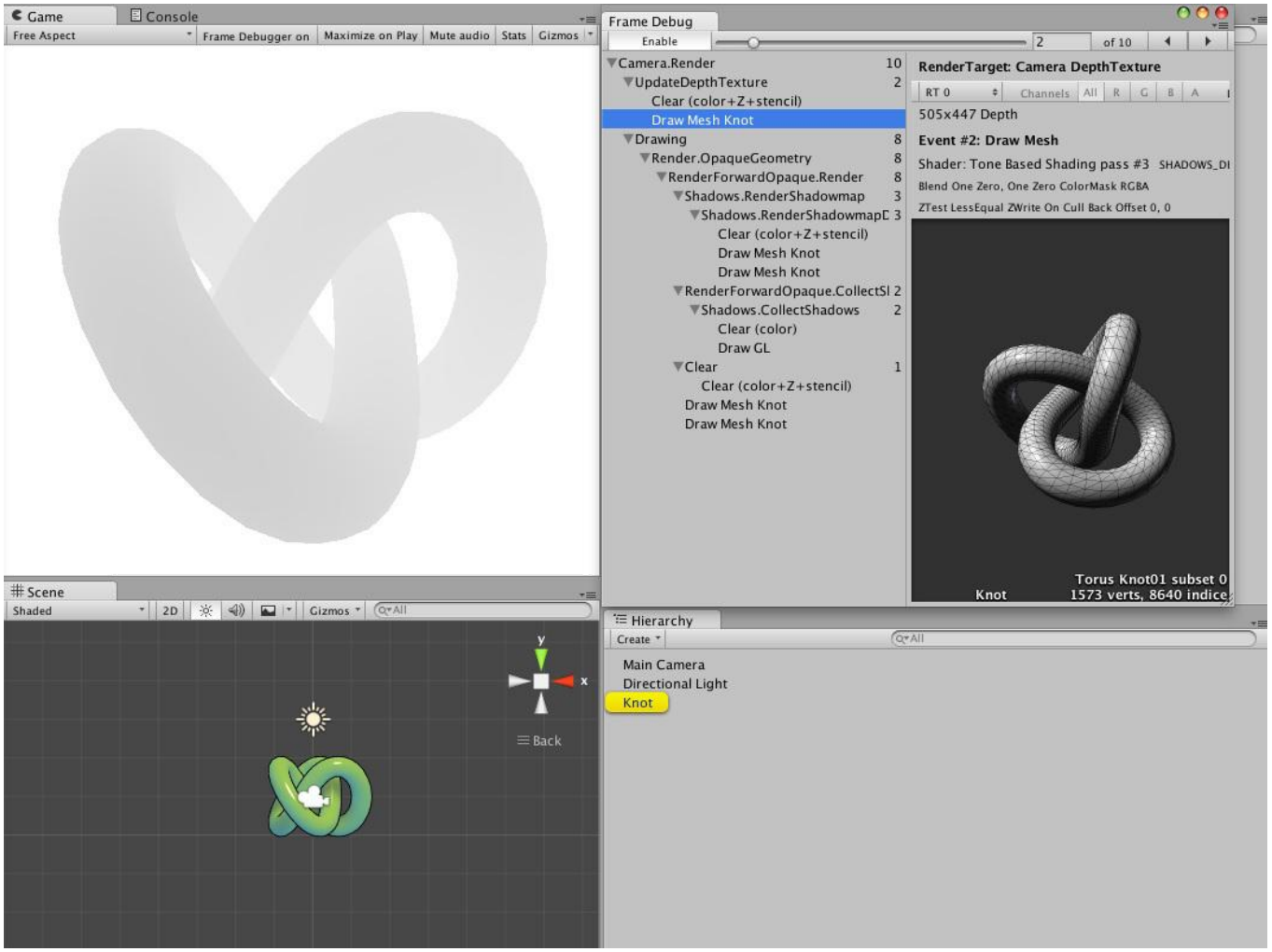
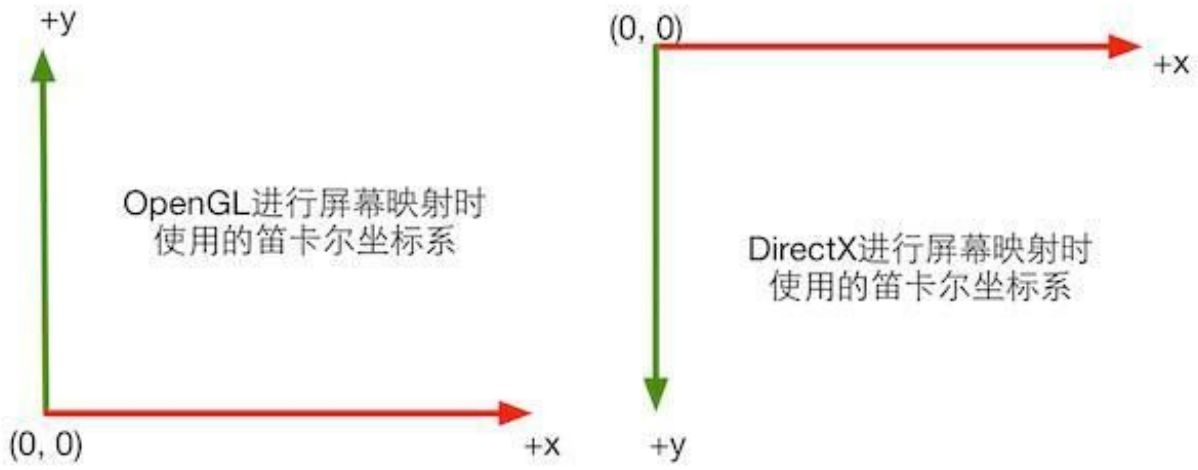

Game
Console
Free Aspect
Frame Debugger on
Maximize on Play
Mute audio
Stats
Gizmos
Frame Debug
Enable
2
of 10
Camera.Render 10
UpdateDepthTexture 2
Clear (color+Z+stencil)
Draw Mesh Knot
Drawing 8
Render.OpaqueGeometry 8
RenderForwardOpaque.Render 8
Shadows.RenderShadowmap 3
Shadows.RenderShadowmapD 3
Clear (color+Z+stencil)
Draw Mesh Knot
Draw Mesh Knot
RenderForwardOpaque.CollectSl 2
Shadows.CollectShadows 2
Clear (color)
Draw GL
Clear 1
Clear (color+Z+stencil)
Draw Mesh Knot
Draw Mesh Knot
RenderTarget: Camera DepthTexture
RT 0
Channels
All
R
G
B
A
505x447 Depth
Event #2: Draw Mesh
Shader: Tone Based Shading pass #3
SHADOWS_DI
Blend One Zero, One Zero ColorMask RGBA
ZTest LessEqual ZWrite On Cull Back Offset 0, 0
Knot
Torus Knot01 subset 0
1573 verts, 8640 indice
Scene
Shaded
2D
Gizmos
All
y
x
Back
Hierarchy
Create
All
Main Camera
Directional Light
Knot

Figure 5.7 click Knot depth map rendering event, display the event in the Game view will display Knot objects in the Hierarchy view will be highlighted, shows a detailof the eventon the right side of the window frame



view5.8 OpenGL and DirectX using a different screen space coordinates

based light inChapter 6 Unity

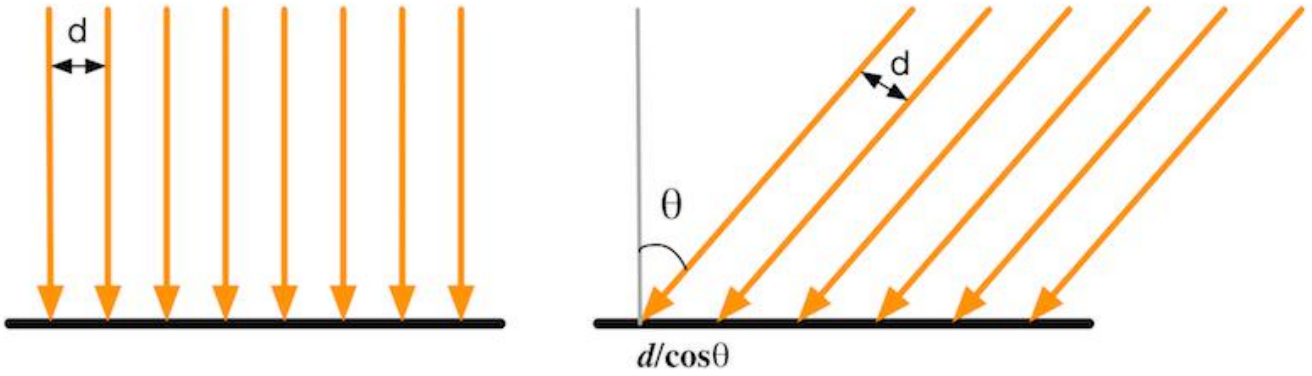
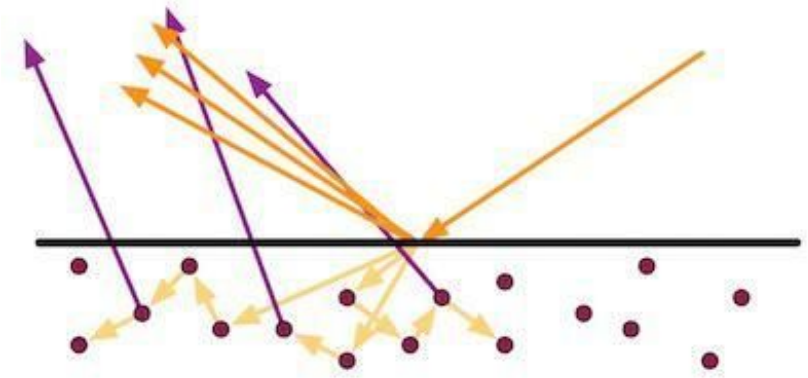


Fig6.1on the left, the light irradiated to the vertical surface of the objectdebugger,Thus the vertical distance between the light remains unchanged; in the right image, light is obliquely irradiated onto the object surface, the distance between the surface of the object light is d / cosθ, because number of received light per unit area on this left to less than



The6.2 scattering, light refraction and reflection phenomenon occurs. For opaque objects, refracted light will continue to propagate inside the body, the final part of the light will again from the surface to be emitted

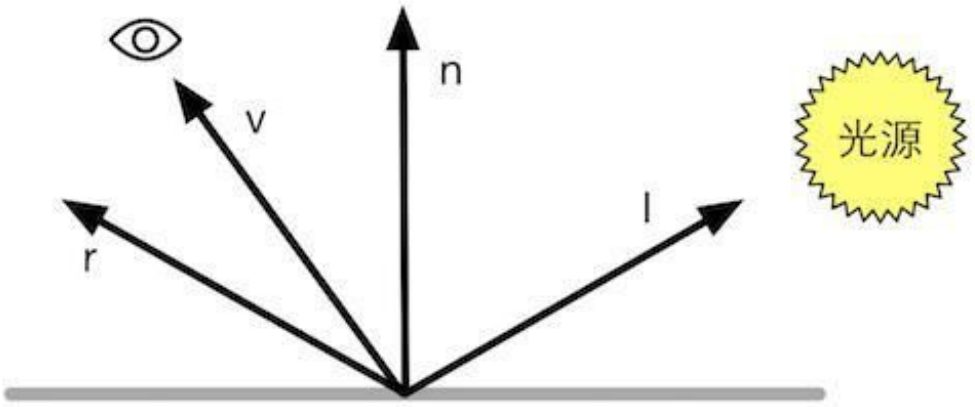


FIG 6.3 Phong model for calculating specular reflection

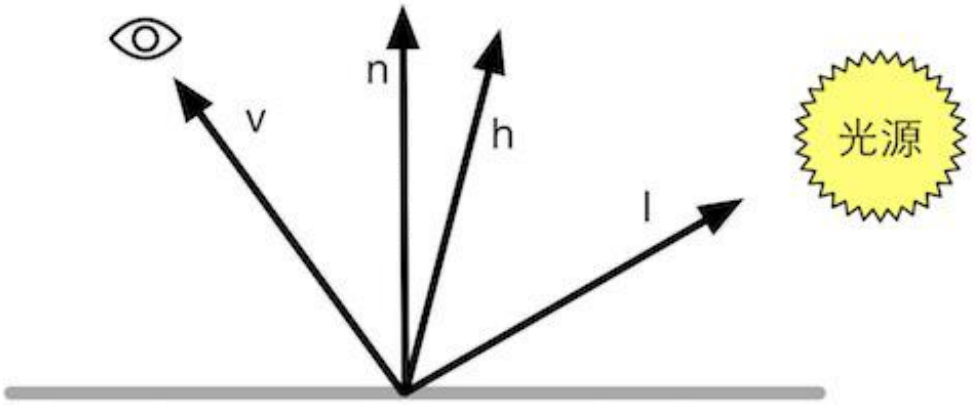


FIG 6.4 Blinn model of

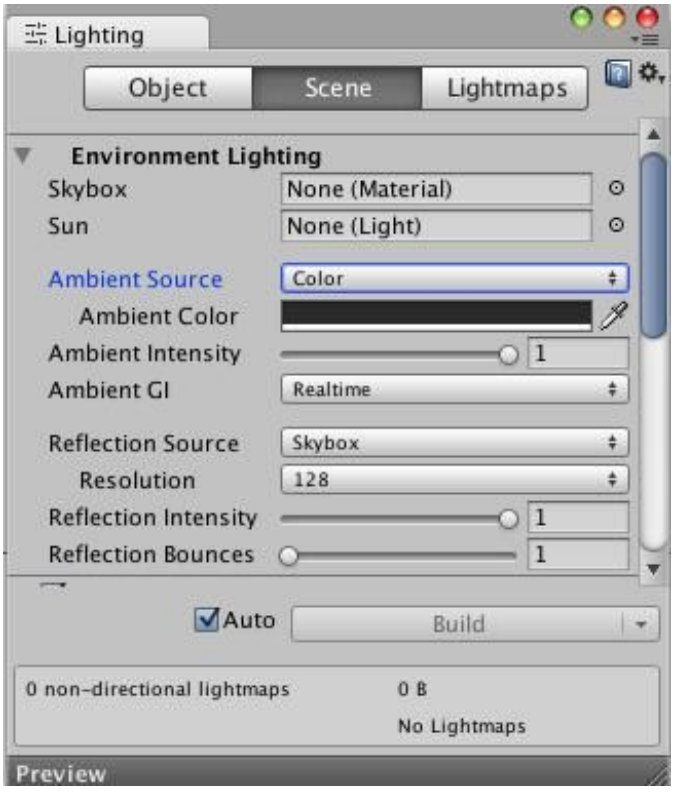
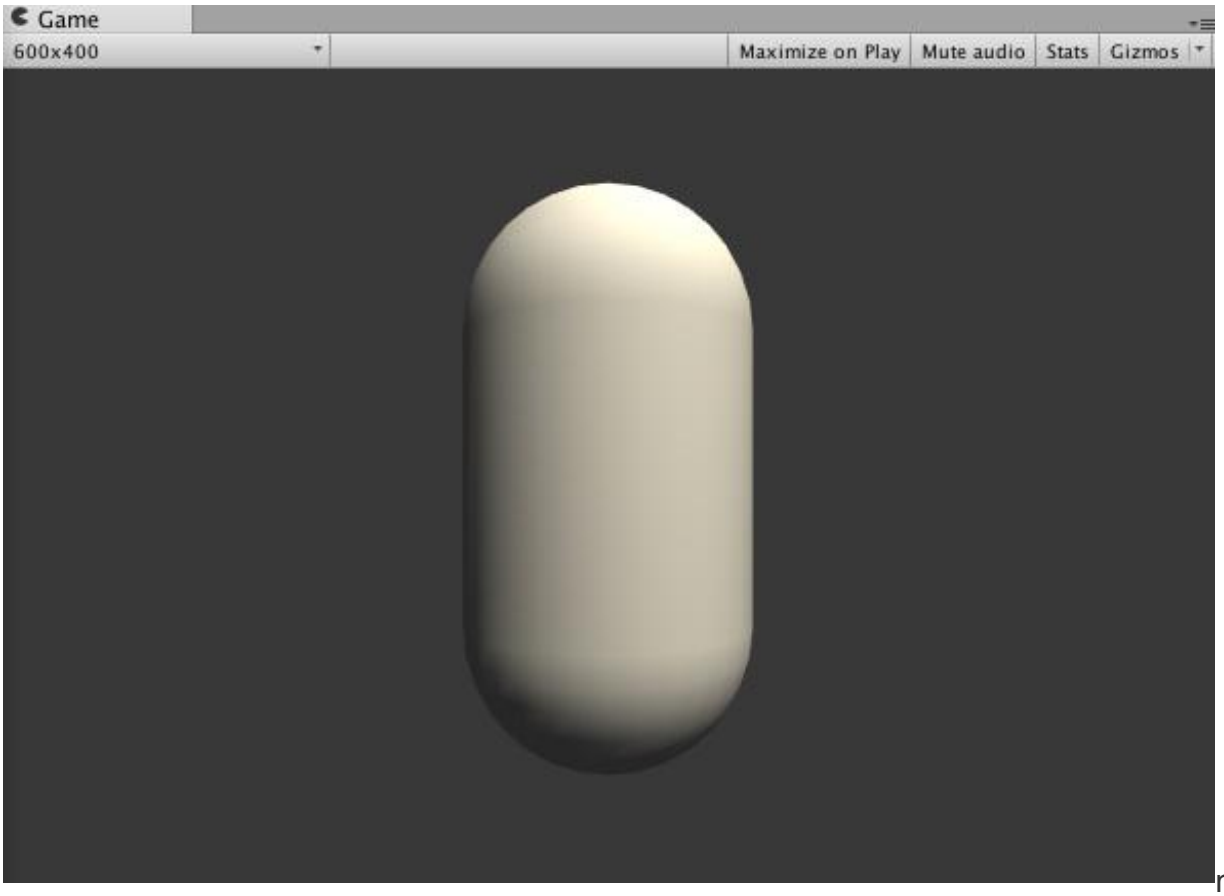


FIG. 6.5 Window in the Unity -> Lighting panel, we can control the color of the ambient light in the scene and intensityby ambient Source / ambient color / ambient intensity



map

6.6 per-vertex diffuse lighting effects

FIG 6.7 pixel-wise diffuse lighting effects

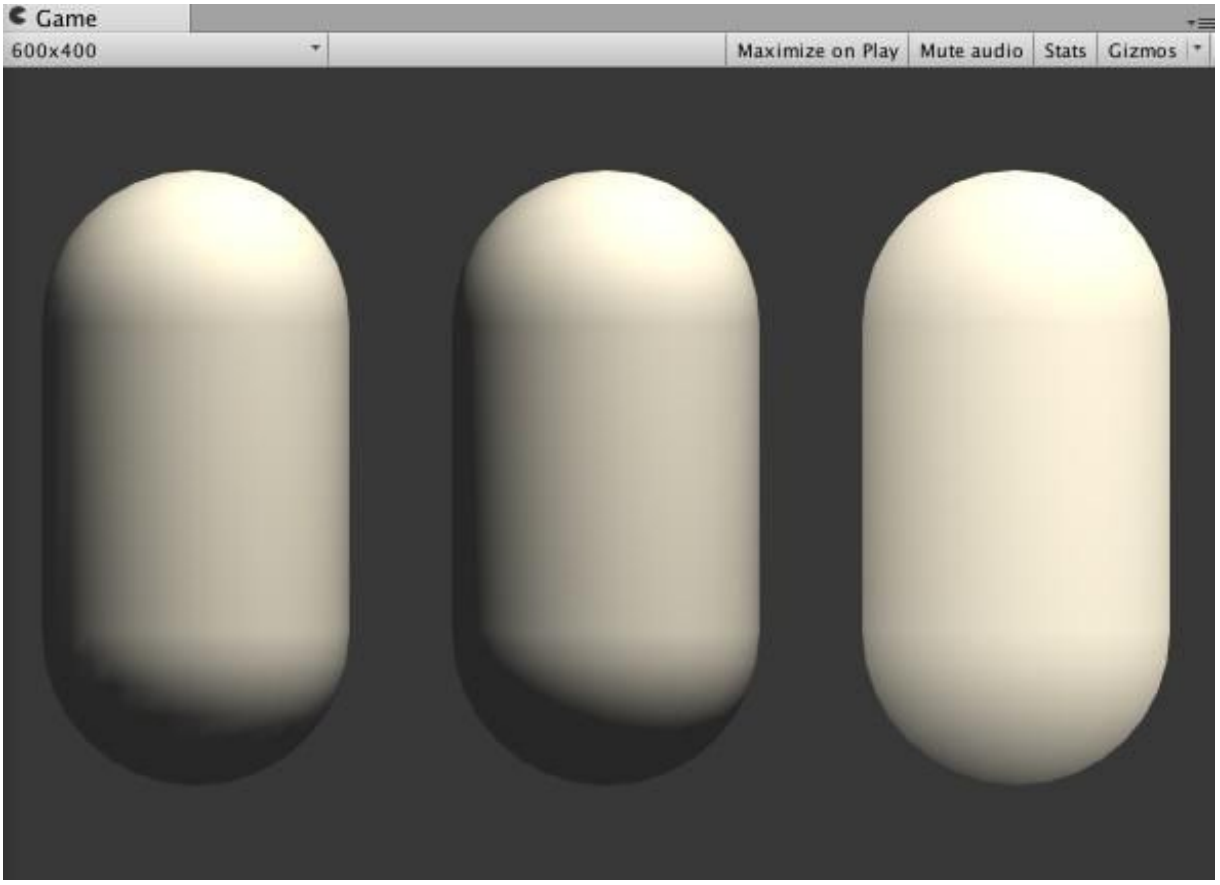
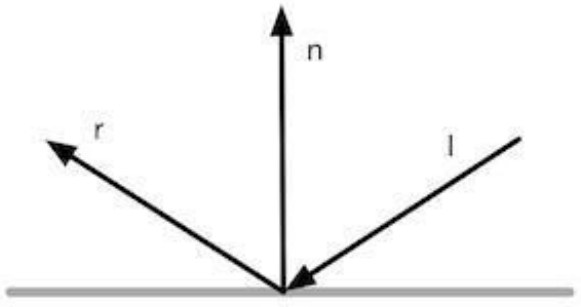


FIG. 6.8-vertex diffuse lightingpixel by pixel, diffuse lighting,contrasthalf Lambert illumination



the reflect function ofFIG6.9 CG

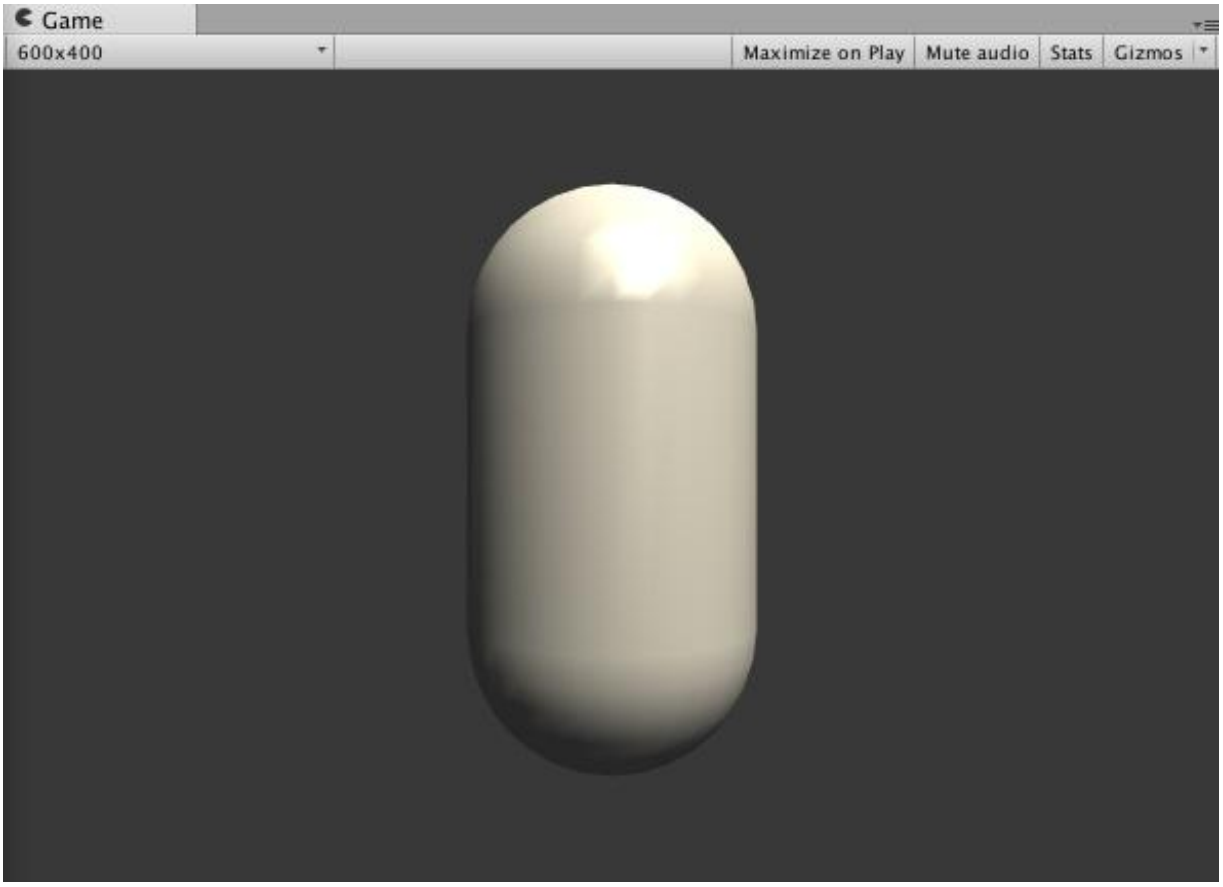
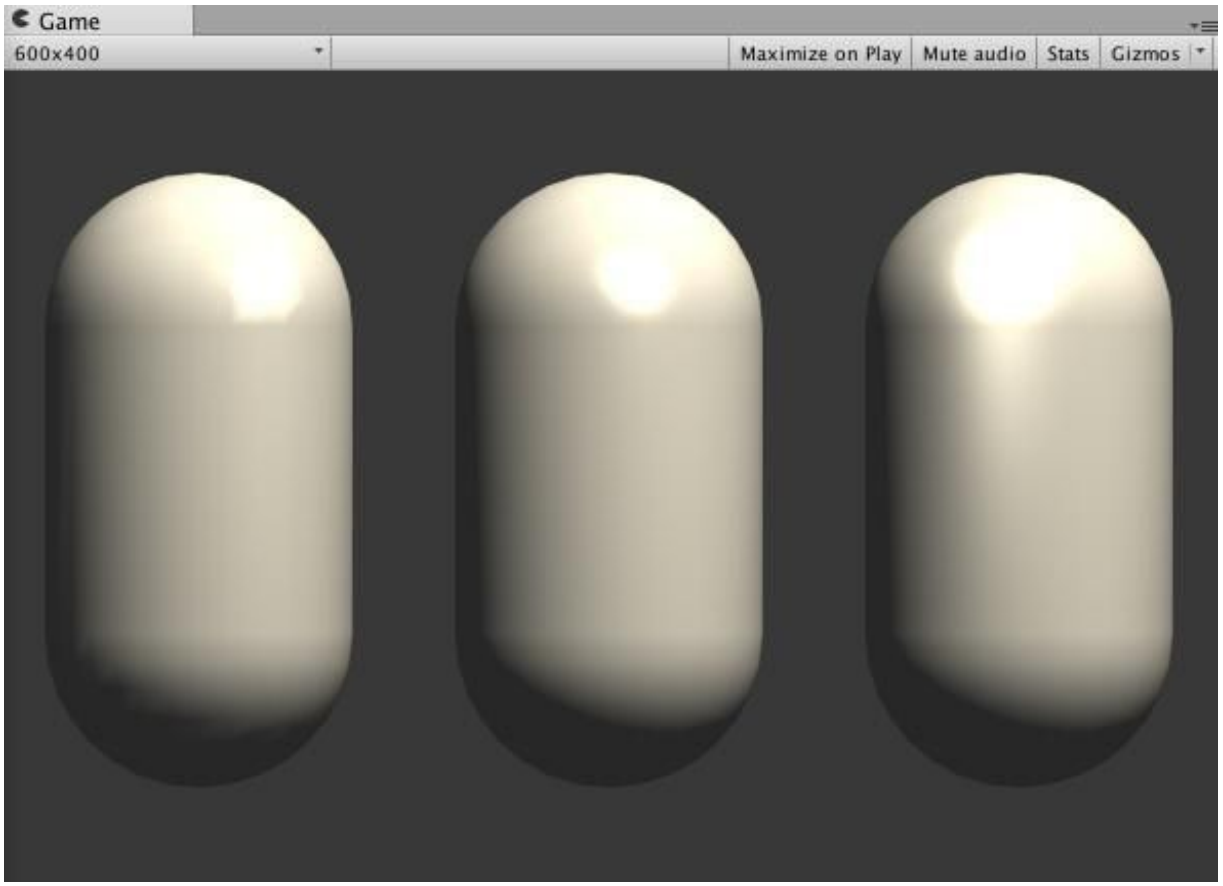


FIG6.10per vertex of specular reflection light

renderings6.11 pixel by pixel specular reflection light



renderings6.12 per vertex of high light reflectance light, pixel by pixel high light reflectance illumination (Phong illumination model) and comparing the results Blinn-Phong illumination high light reflectivity

Chapter 7 base texture

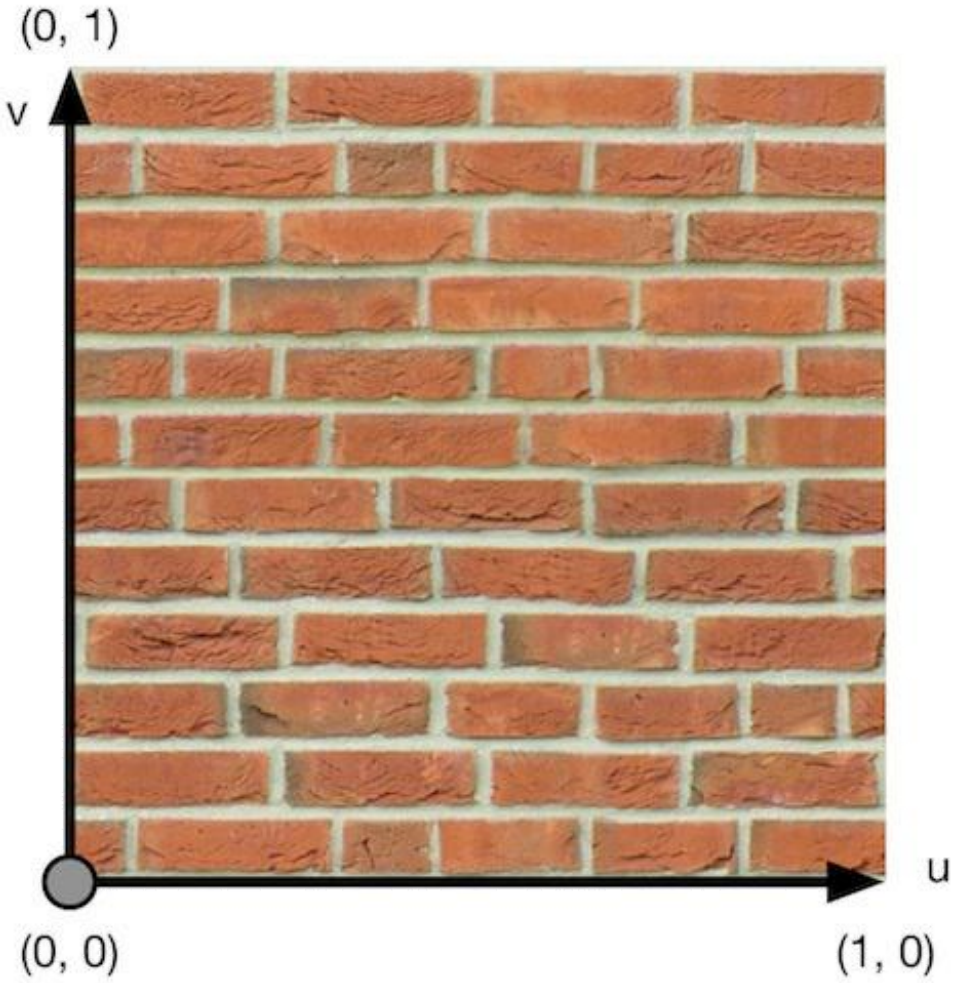


Figure 7.1texture coordinates Unityin

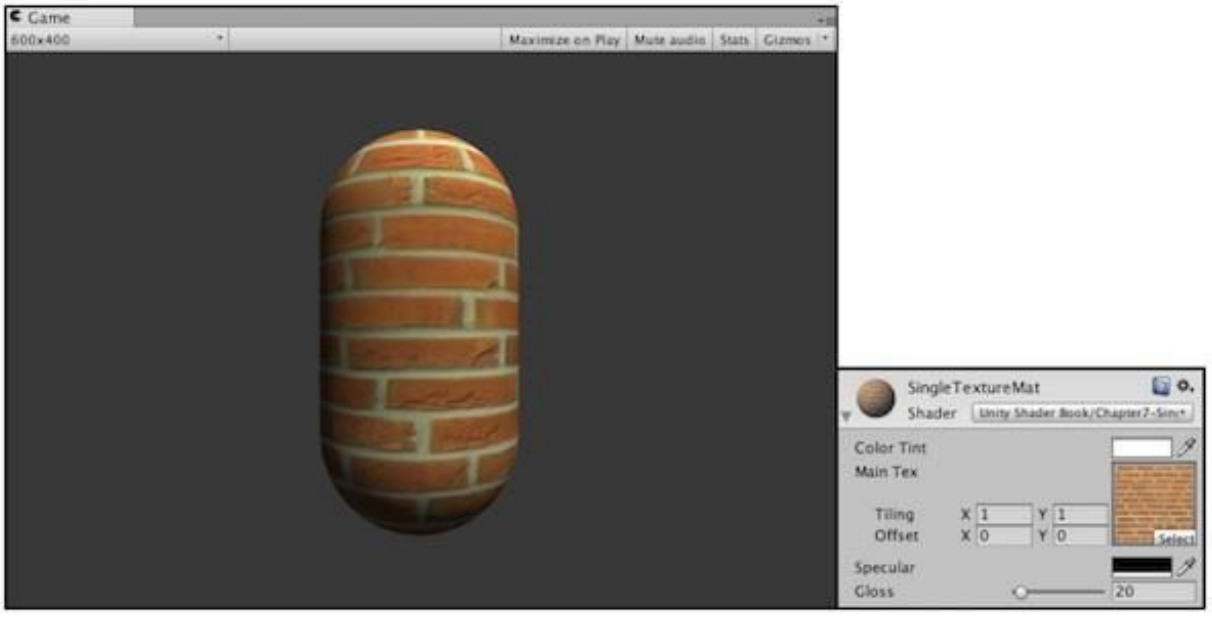
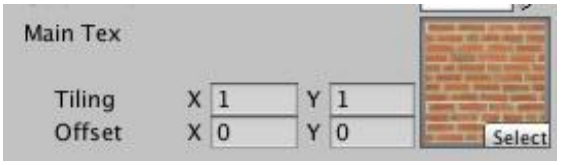


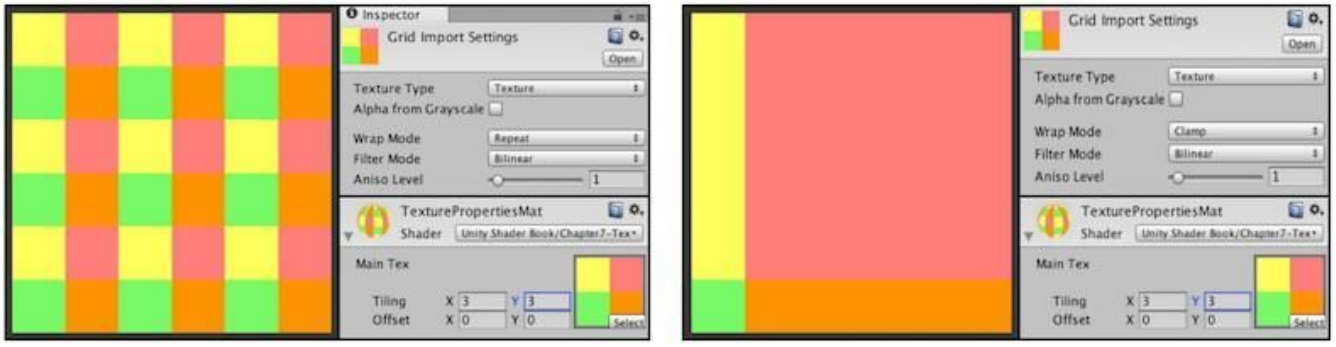
Figure 7.2 using thesingle texture

maptexture tiled 7.3 offset



adjustment (zoom) and (translation)properties

FIG texture7.4attribute

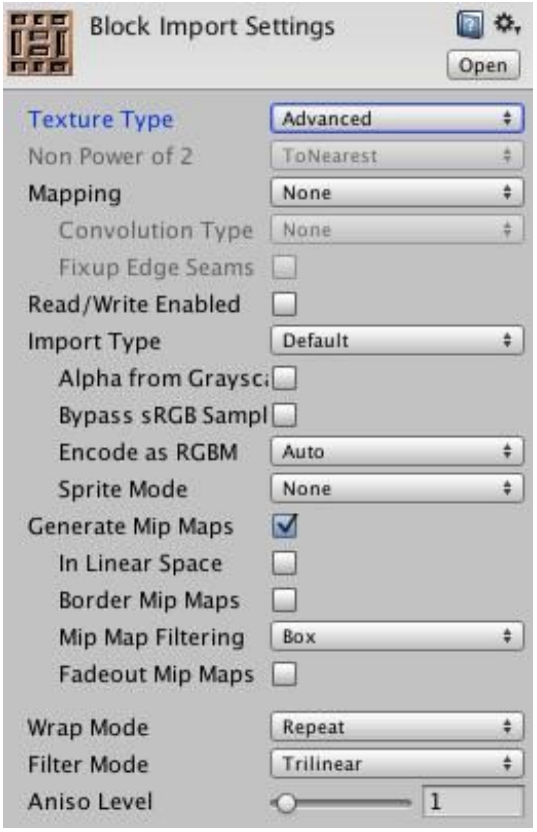


7.5 Wrap Mode FIGdetermines if the texture coordinates over [0, 1] After aroundtile will be how the

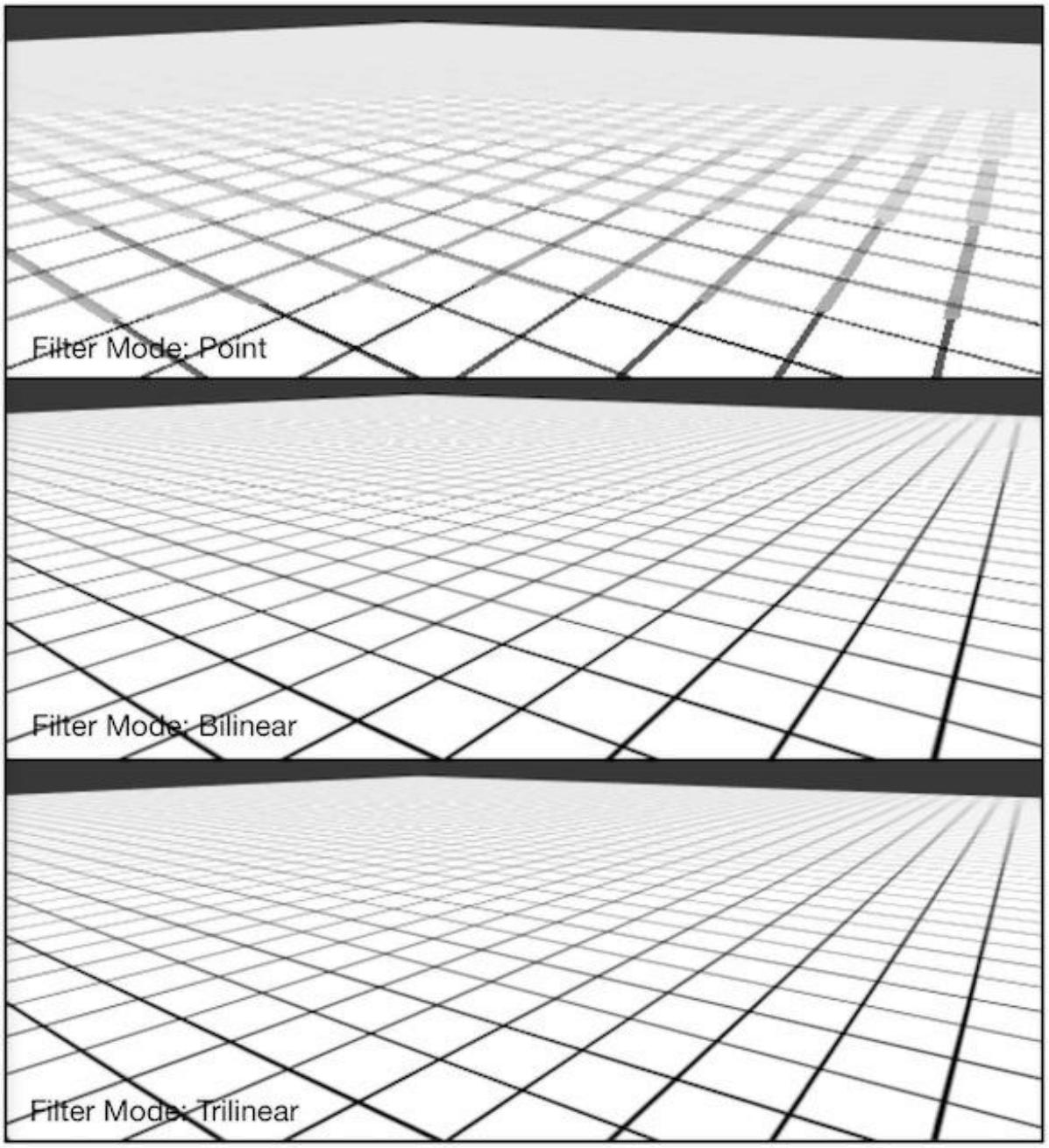
7.6 tooffset (Offset) determines the shift amountattribute texture coordinates

Filter Mode: Point　　Filter Mode: Bilinear　　Filter Mode: Trilinear

of FIG7.7 during amplification textureofthree results were used to give Filter Mode



views 7.8Advanced mode may be provided in a multi-stage texture attributes related to further away



from top to bottom in FIG. 7.9: Point filter plusmulti-stage further away texturing techniques, Bilinear filtering + multistage further away texturing techniques, Trilinear + multistage filter distancedtexturing technique

the maximum size of 

the7.10selected texture and

FIG.texture pattern

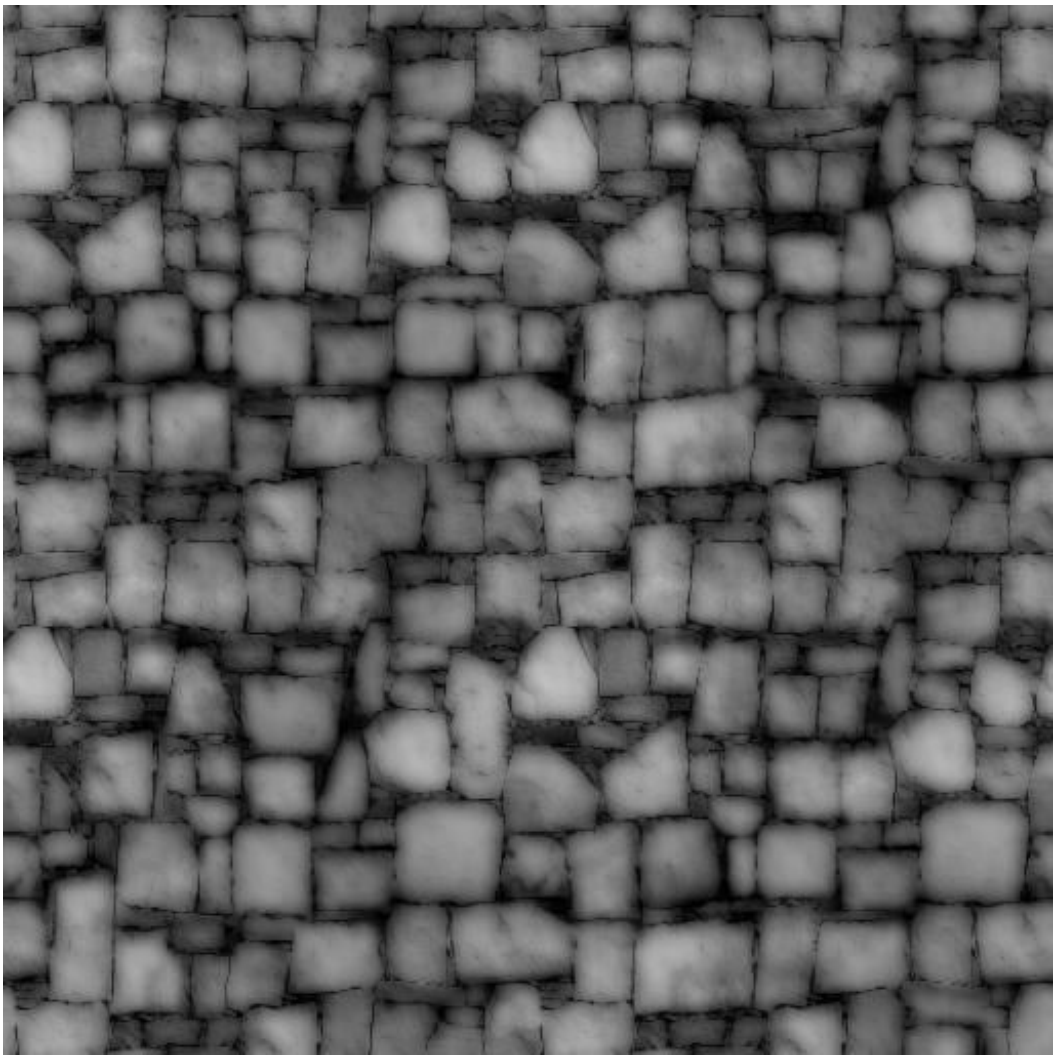
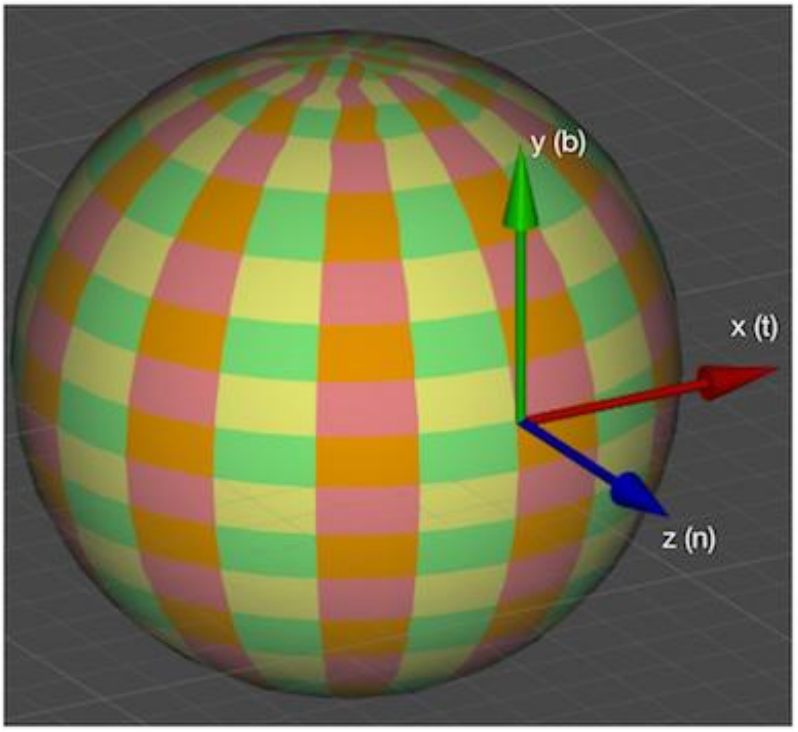


FIG7.11heightFIG.



7.12 tangent spacemodel vertices Wherein the vertex coordinate corresponding to an origin, x-axis is tangential direction (t), y-axis is the sub tangential direction (b), z-axis normal direction (n)

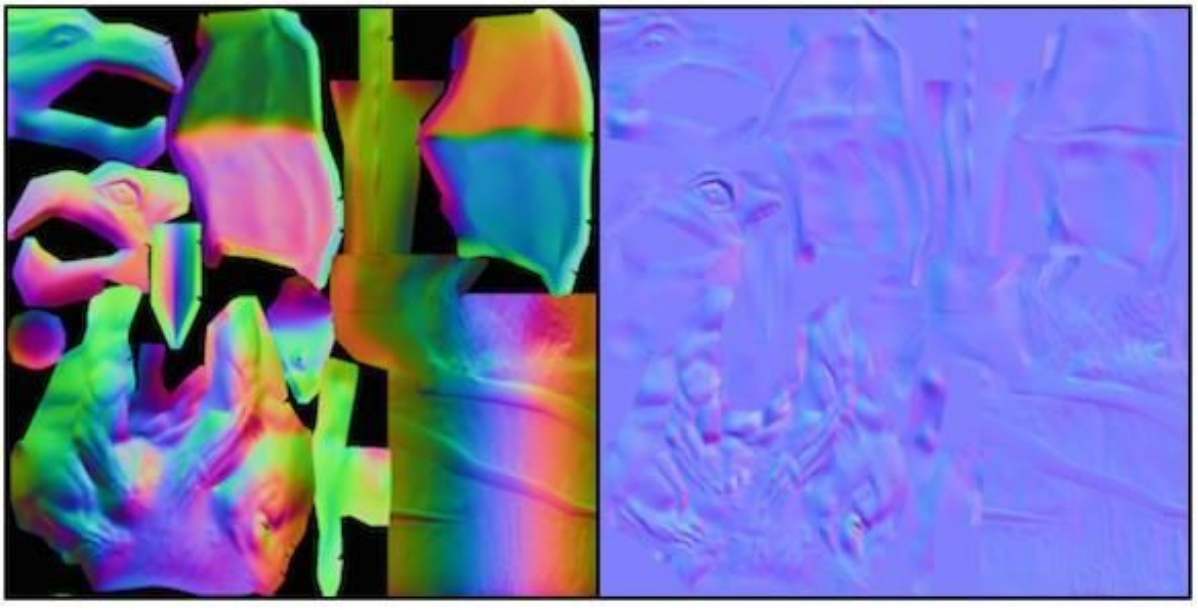
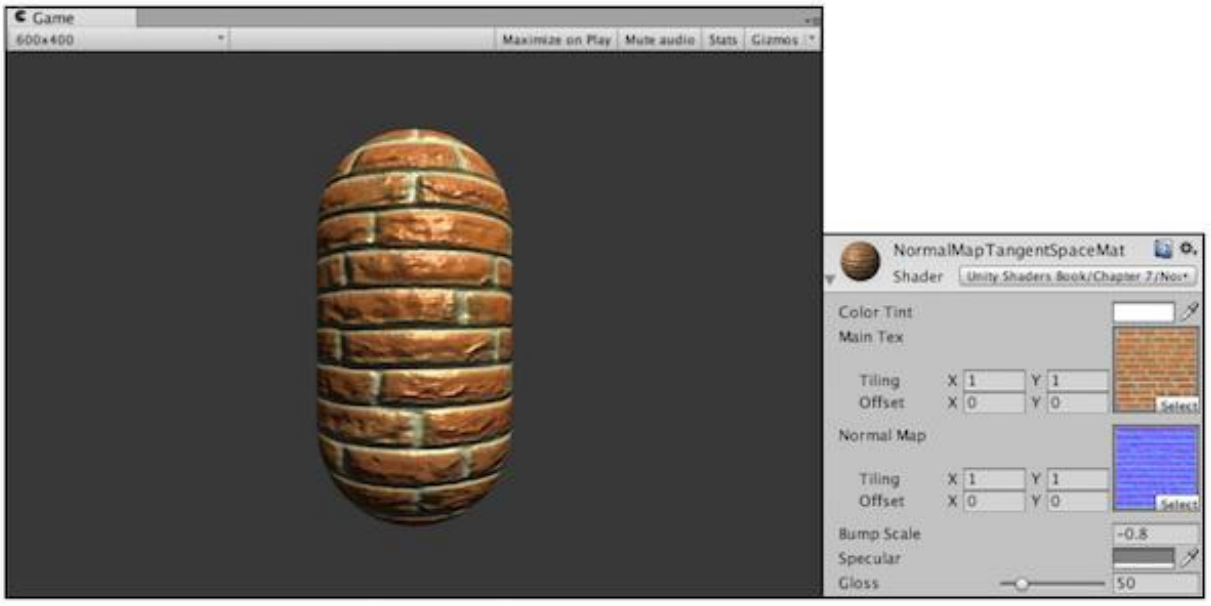


Figure 7.13 Left: normal texture in model space. Right: normal texturein the tangent space



maps usednormal 7.14texture

7.15 usemaps Bump Scale properties to adjust the degree of irregularitiesthe model

7.16 whenof FIG normal direction normal use UnpackNormal texture function, texture type identifier needNormal map is

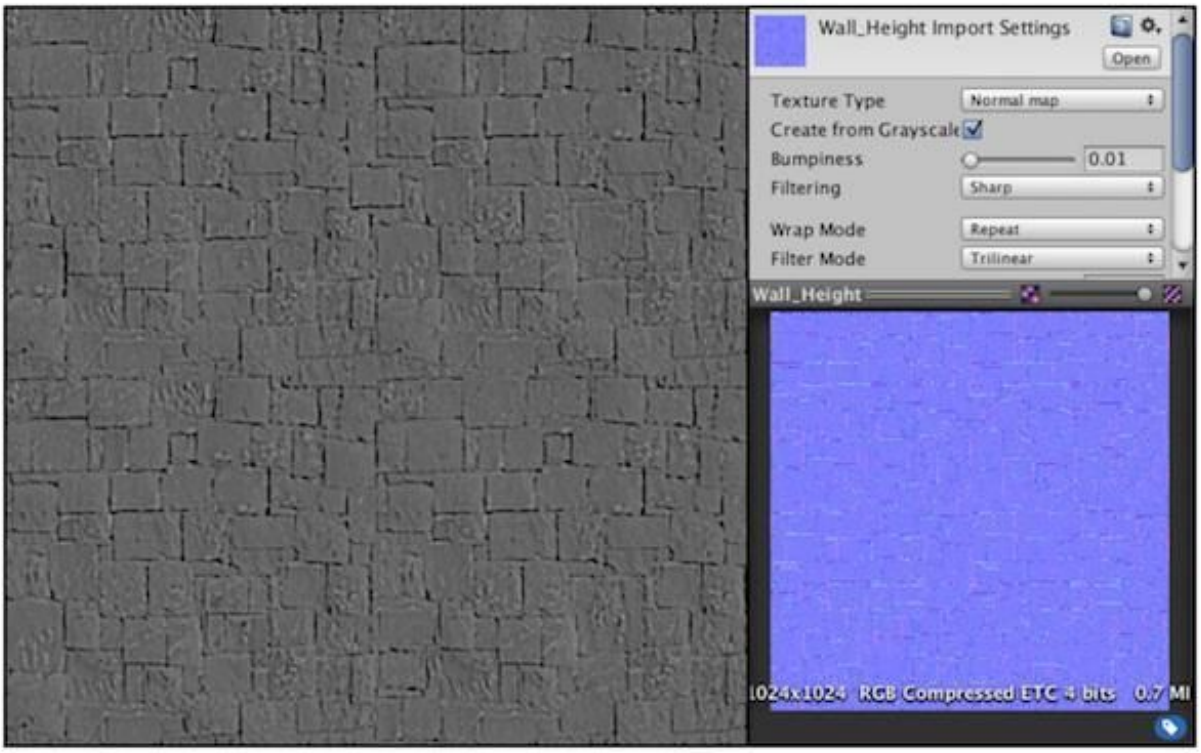
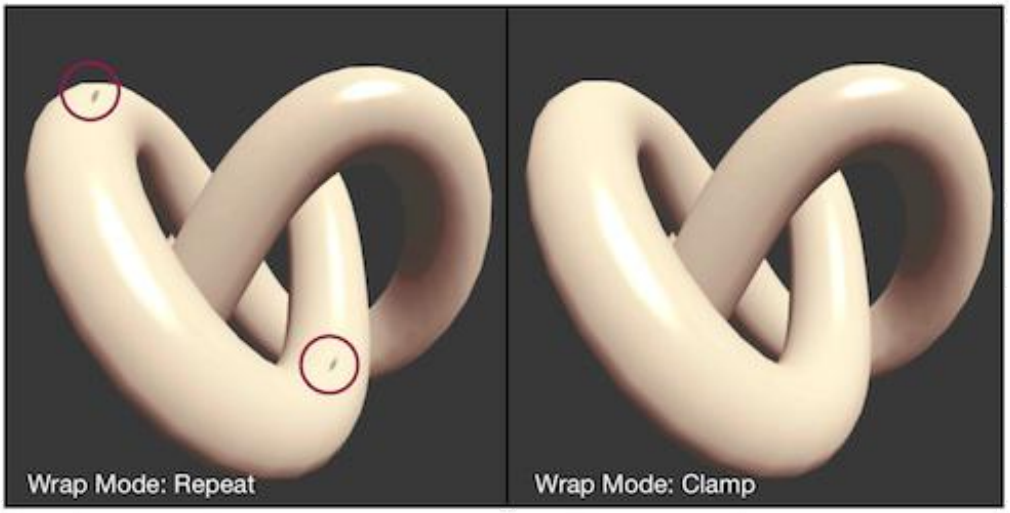


FIG after 7.17checked when the Create from Grayscale, Unity texturewill be generated under a normal tangent spacethe height FIG

mapsusing different gradients 7.18 texture controldiffuse lighting, the lower left corner of each figure shows the ramp textureused

mapwere 

7.19 Wrap mode and effects Repeat mode Clampcomparisonto

7.20chart using a high light texturethemask. From left to right: it contains only diffuse reflection, the mask unused high light reflectance, high light reflectance using a mask of

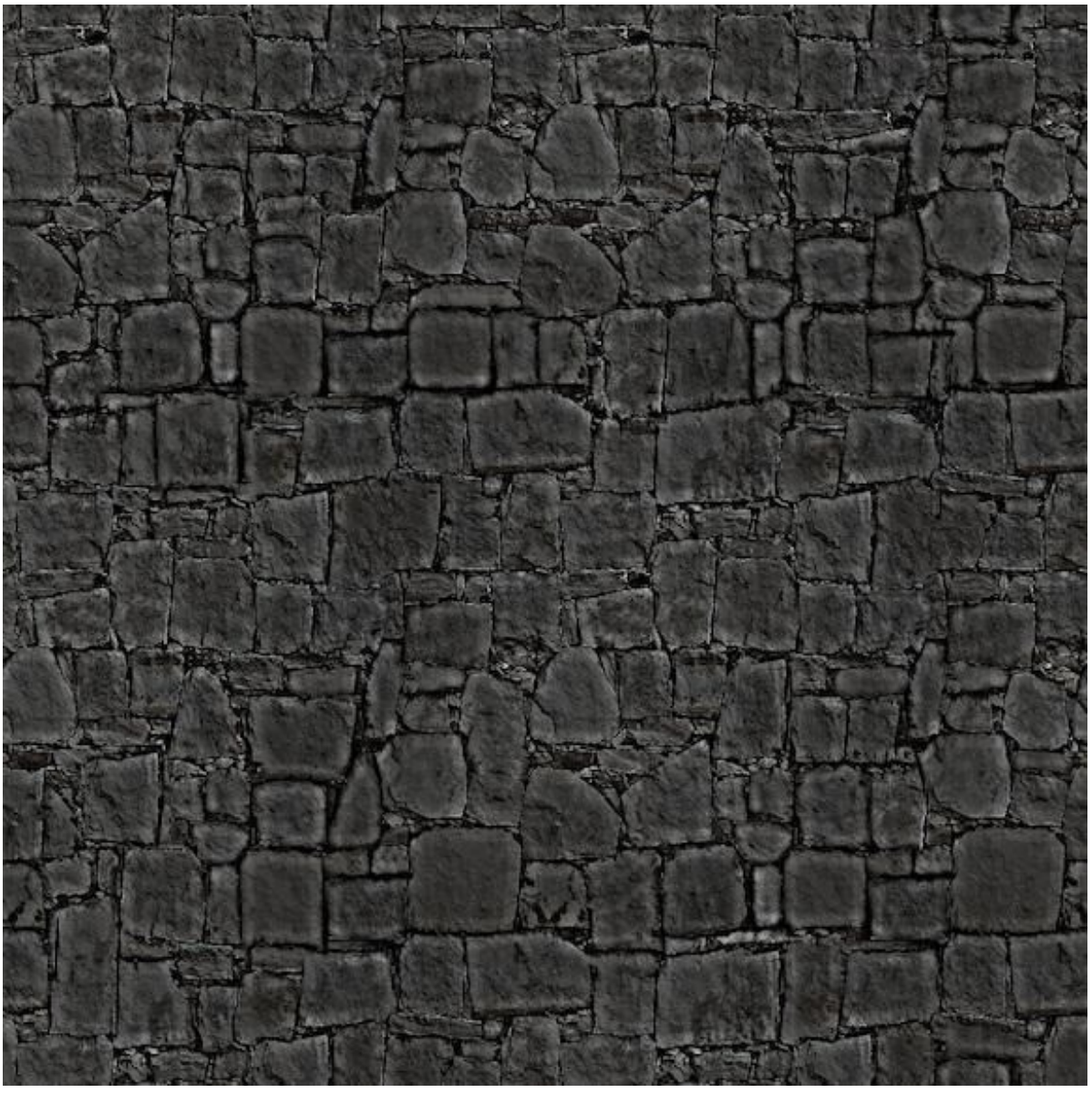
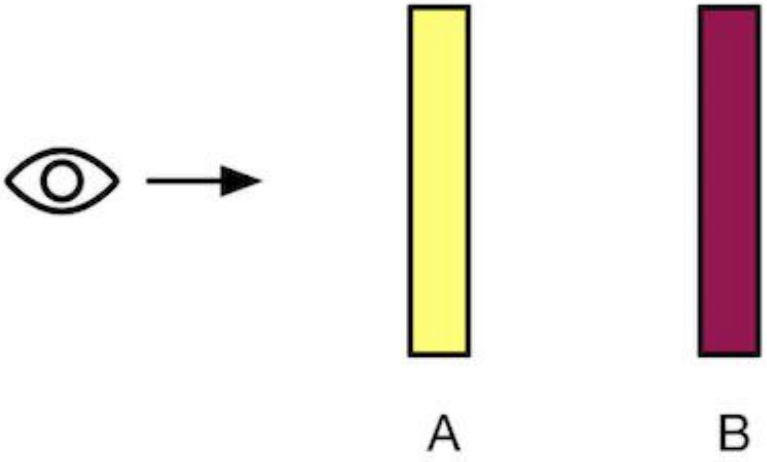


FIG mask texture highlights 7.21section usedthe

Chapter 8renderingtransparent



8.1 scene intwo objects, wherein A (yellow) is translucent objects, B (violet)is an opaque object

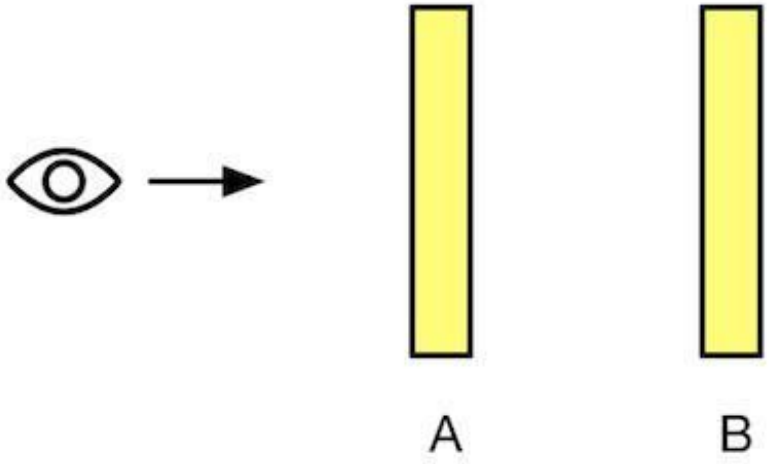


Figure 8.2in the scene with two objects, wherein a and B aretranslucent object

8.3 cycles  FIGoverlapping translucent objects can not always correct translucency

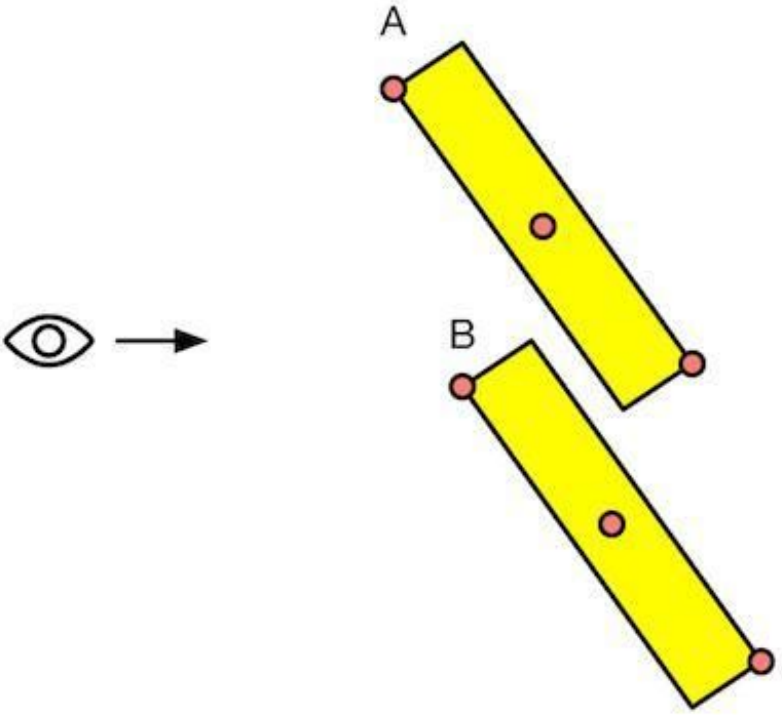
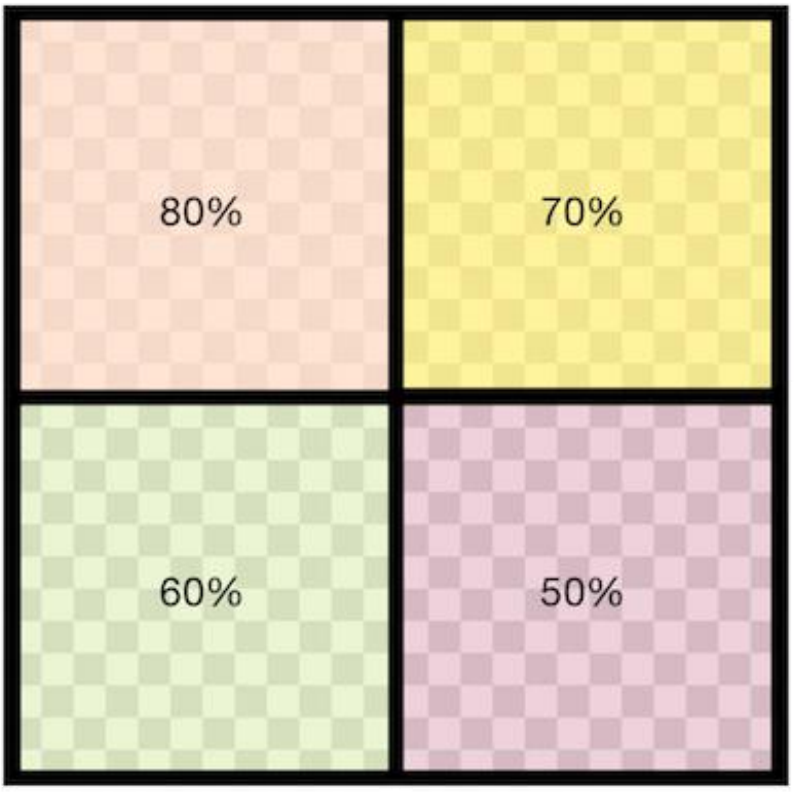


Figure 8.4 using a depth which the object to be sorted. Red points are marked on the grid point nearest the camera, the point farthest grid andmidpoint

a transparent 

textureFigure 8.5, where each square is different transparency

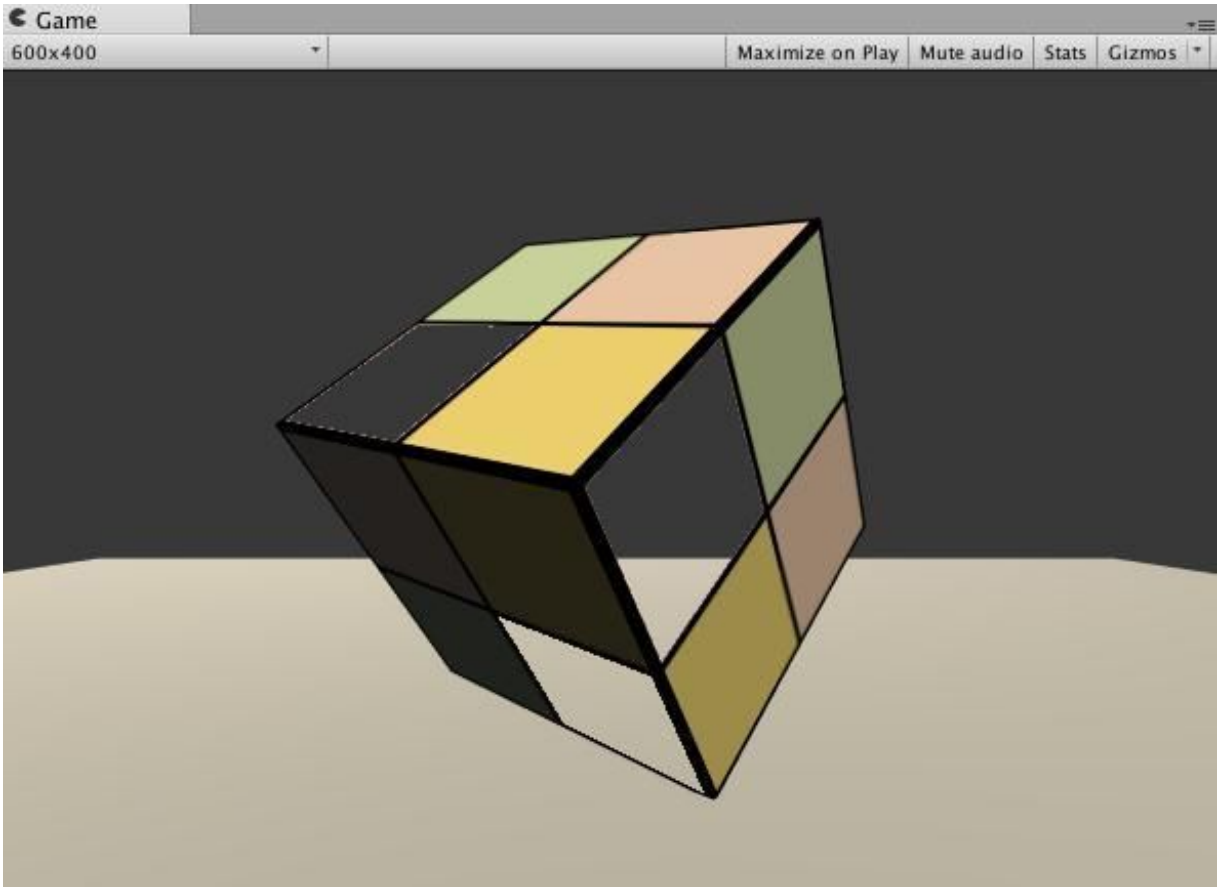


Figure 8.6Clarity Test

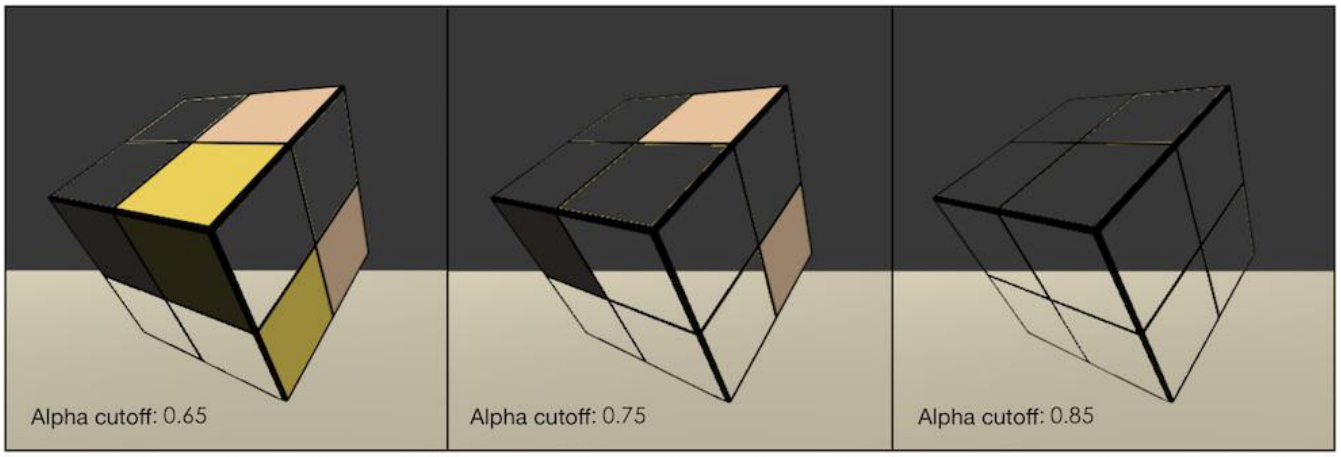


Figure 8.7With Alpha cutoff parameter increases, the more the transparency of the pixel does not satisfy the conditions of the test were excluded

Figure 8.8alpha blending

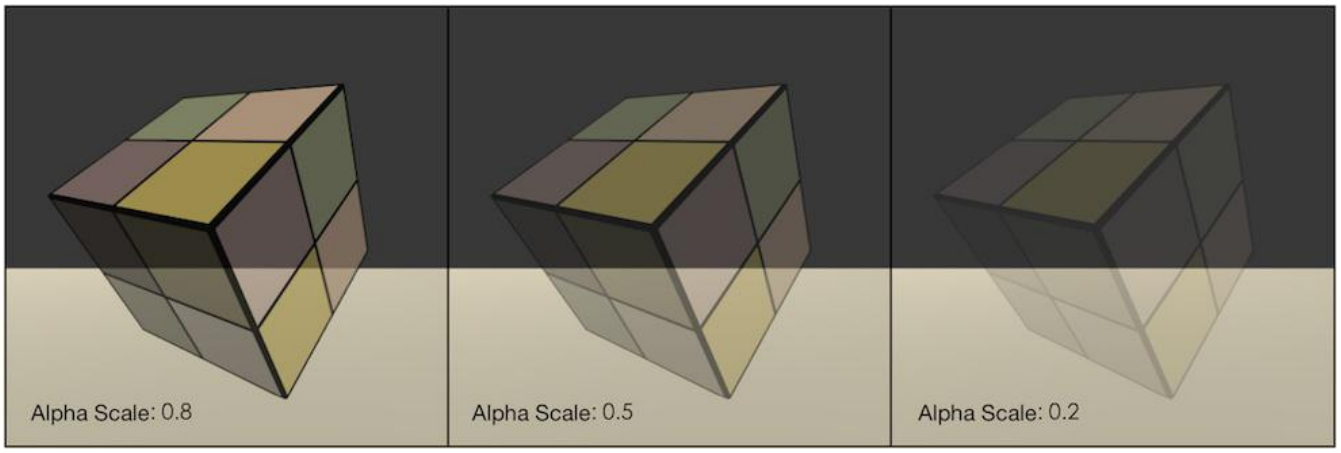
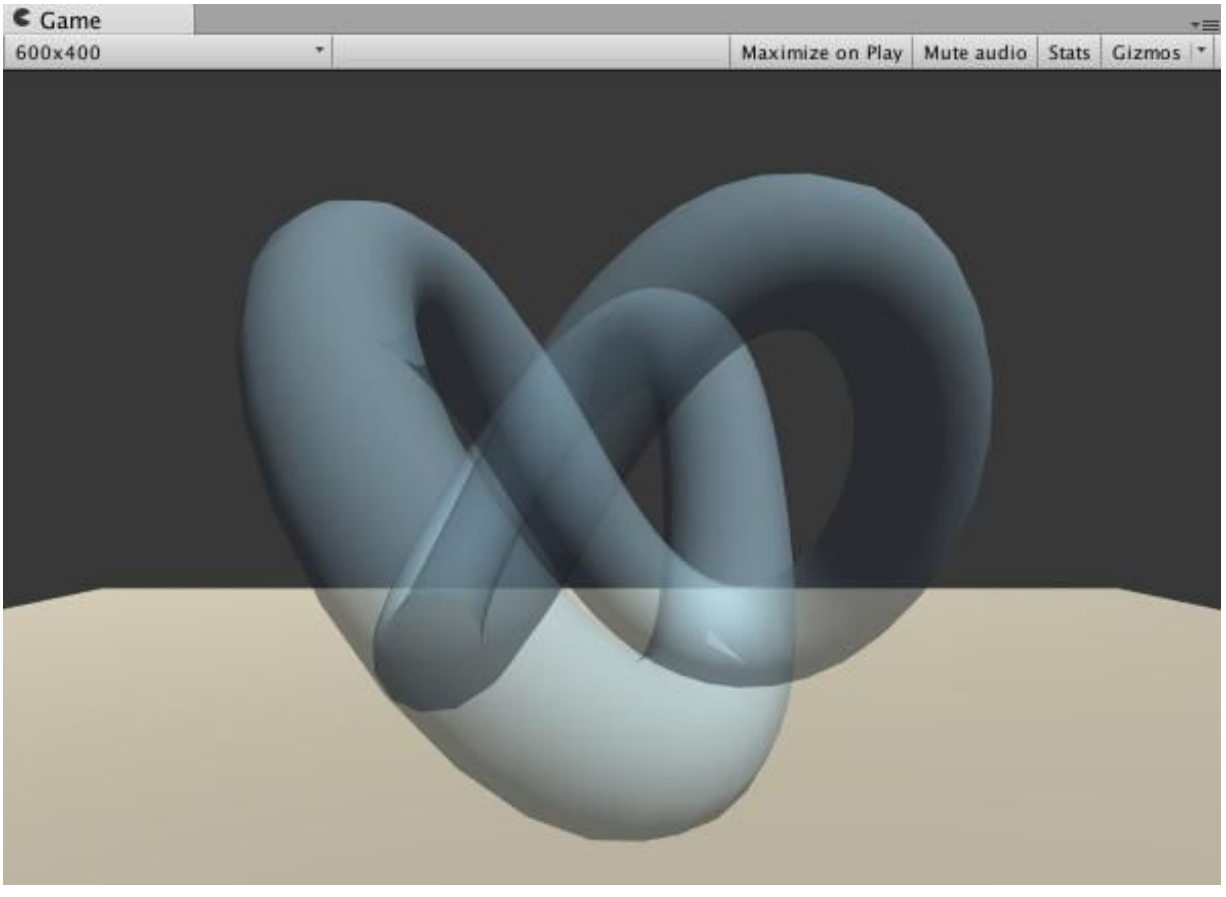
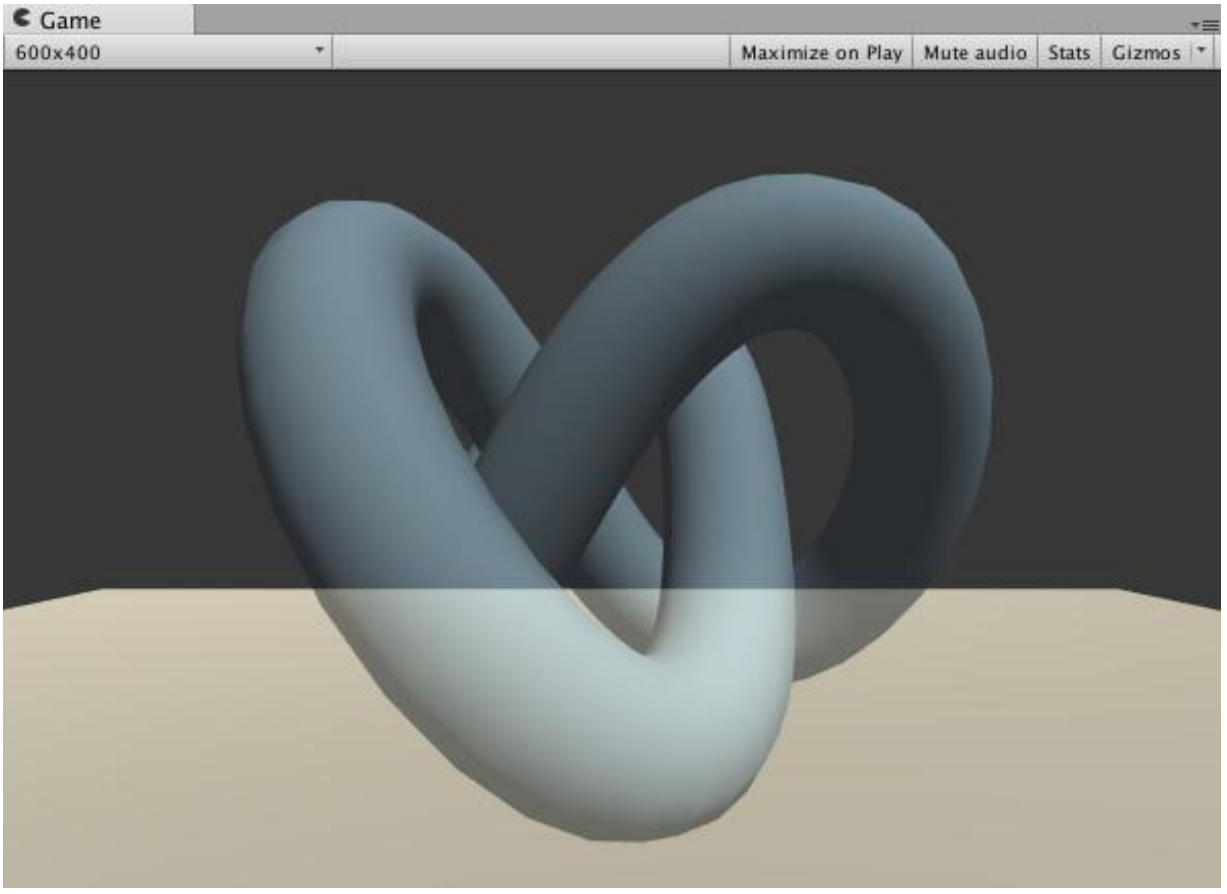


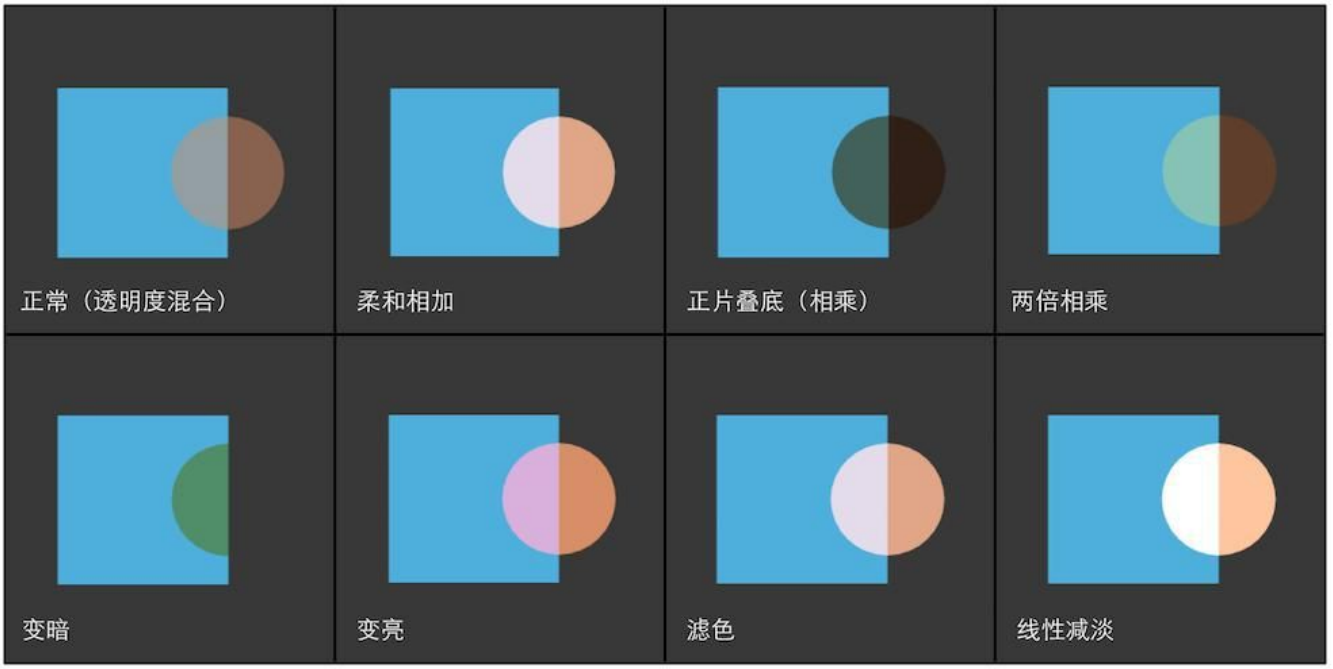
Figure 8.9 with increasingAlpha Scale parameters, the model becomes more transparent



when there is between 8.10 FIG model grid interdigitate when the structure, errors tend to gettranslucency



8.11FIG turned translucentofdepthwriting



effect 8.12different set mixed state obtained

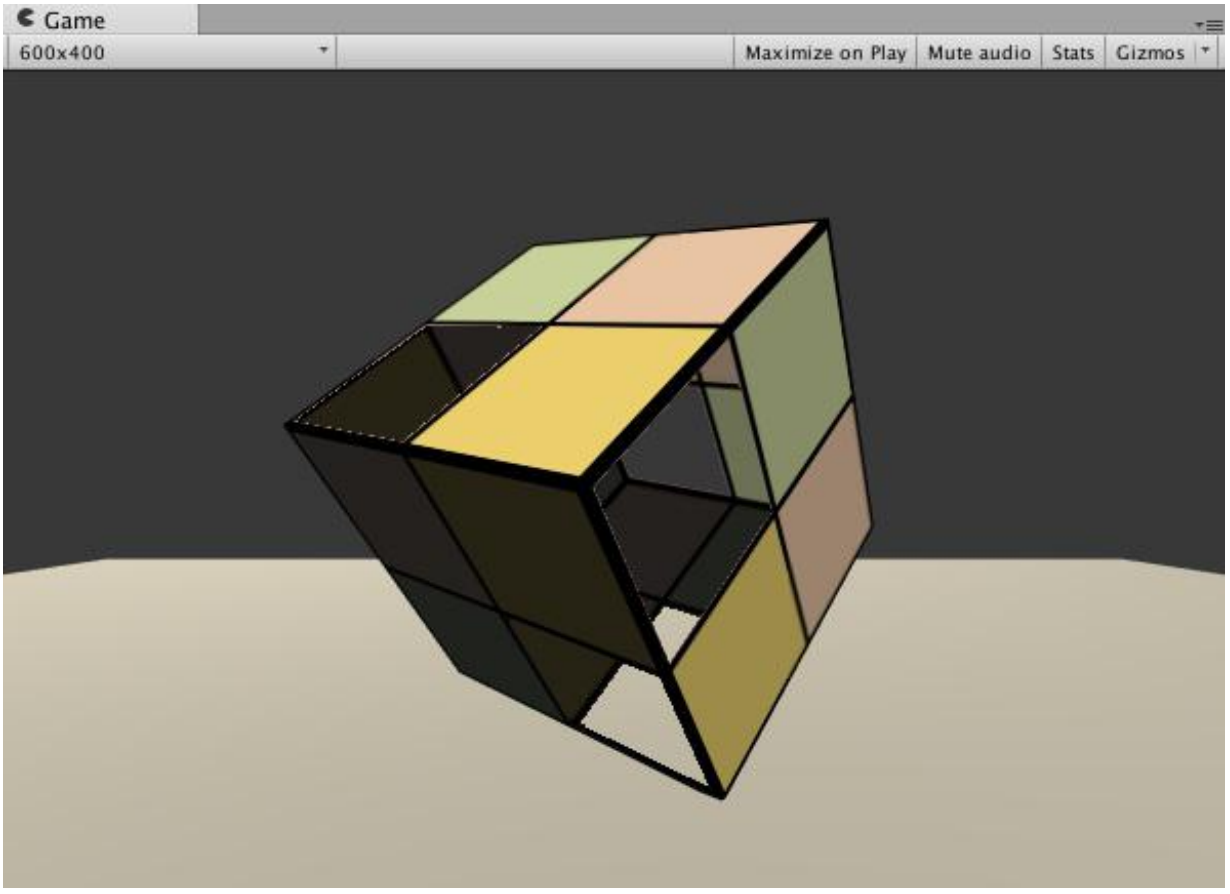
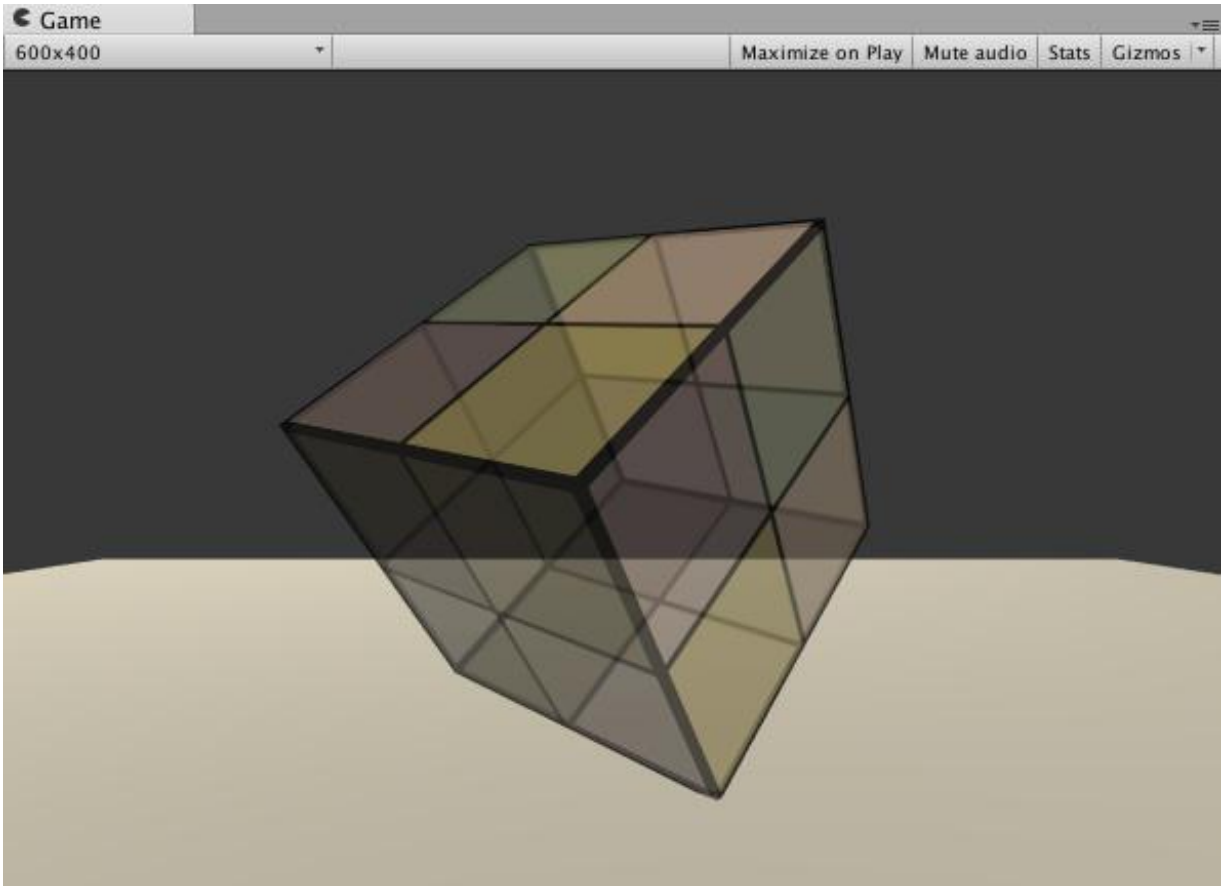


FIG8.13 sided test objecttransparency rendered



mapdouble-sided transparency rendered mixing 8.14 object

Chapter 9 a more complex lighting

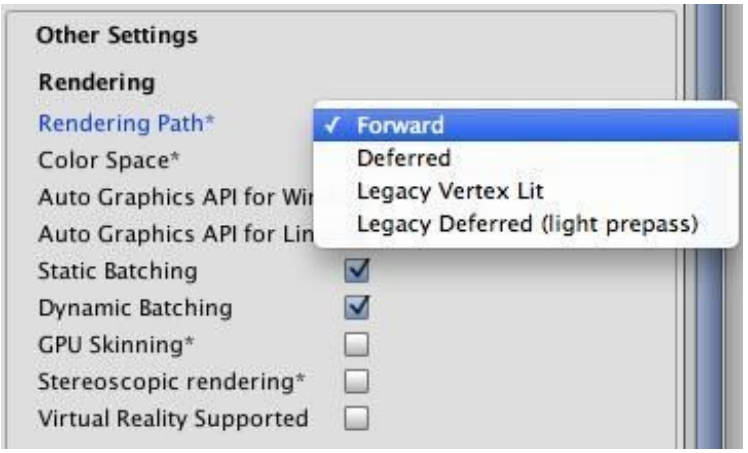
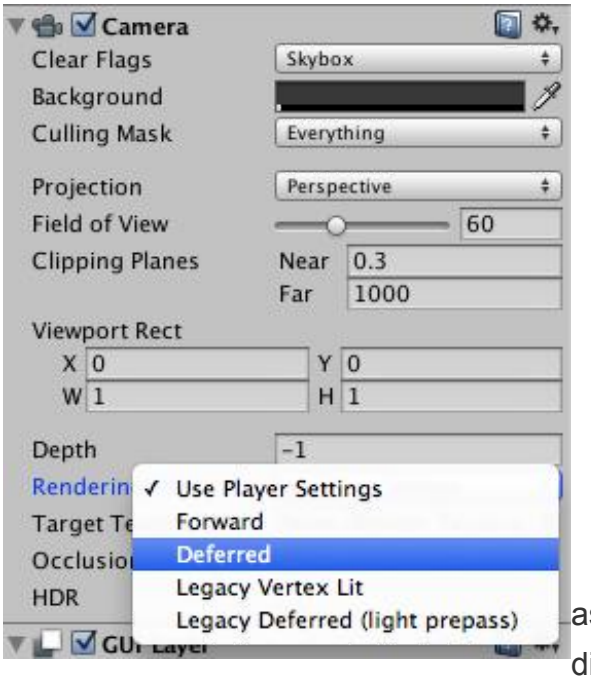


Figure 9.1 is provided rendering pathUnity project

providedFIG9.2 camera 

assembly rendering path may cover disposedproject settings of

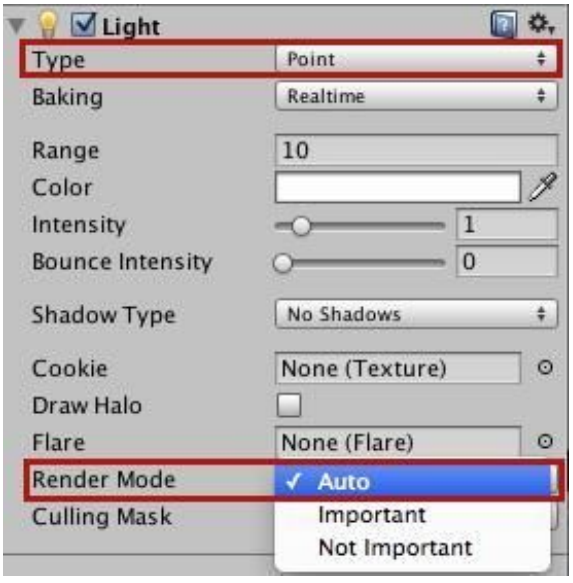


FIG9.3 the light source type and rendering mode

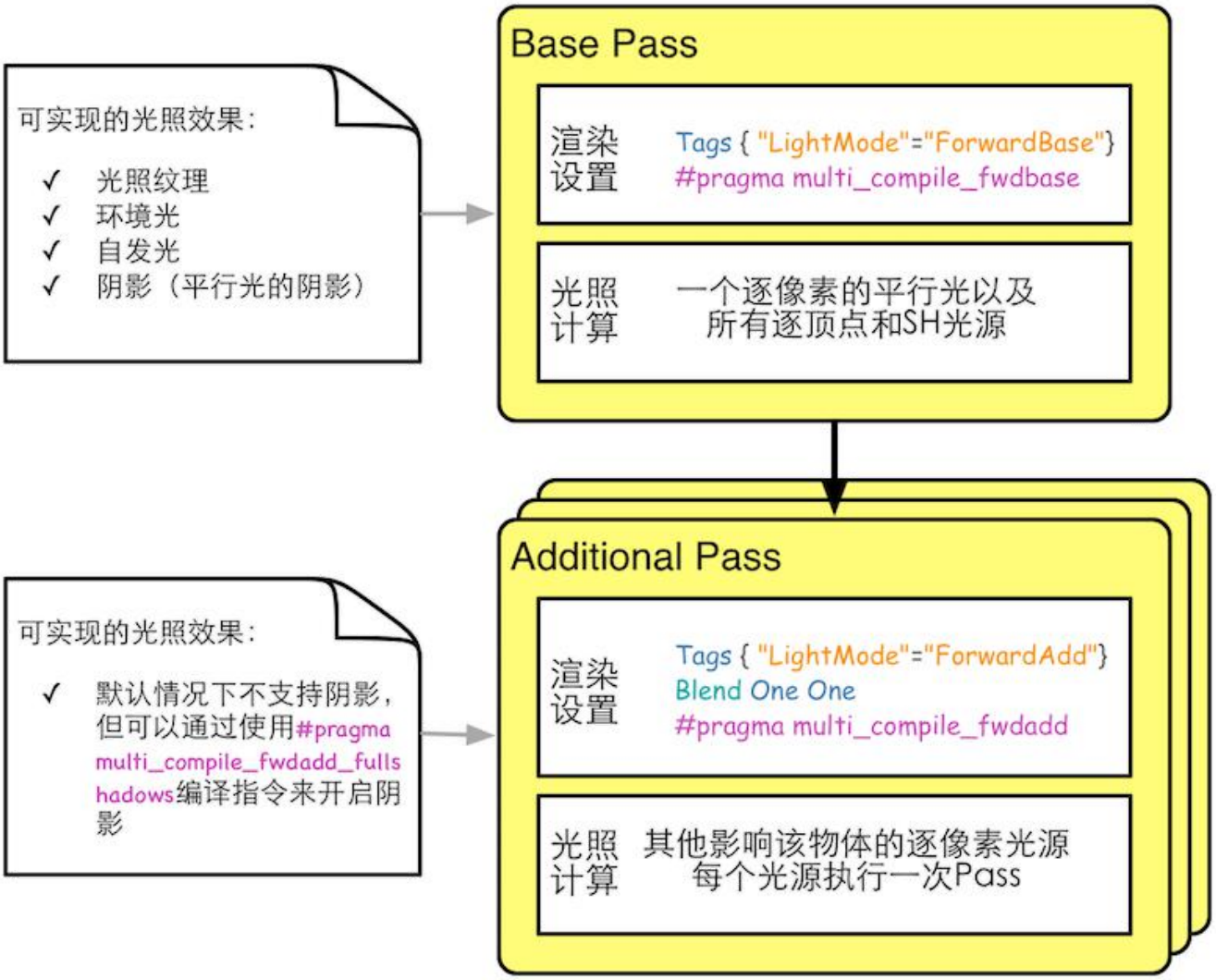
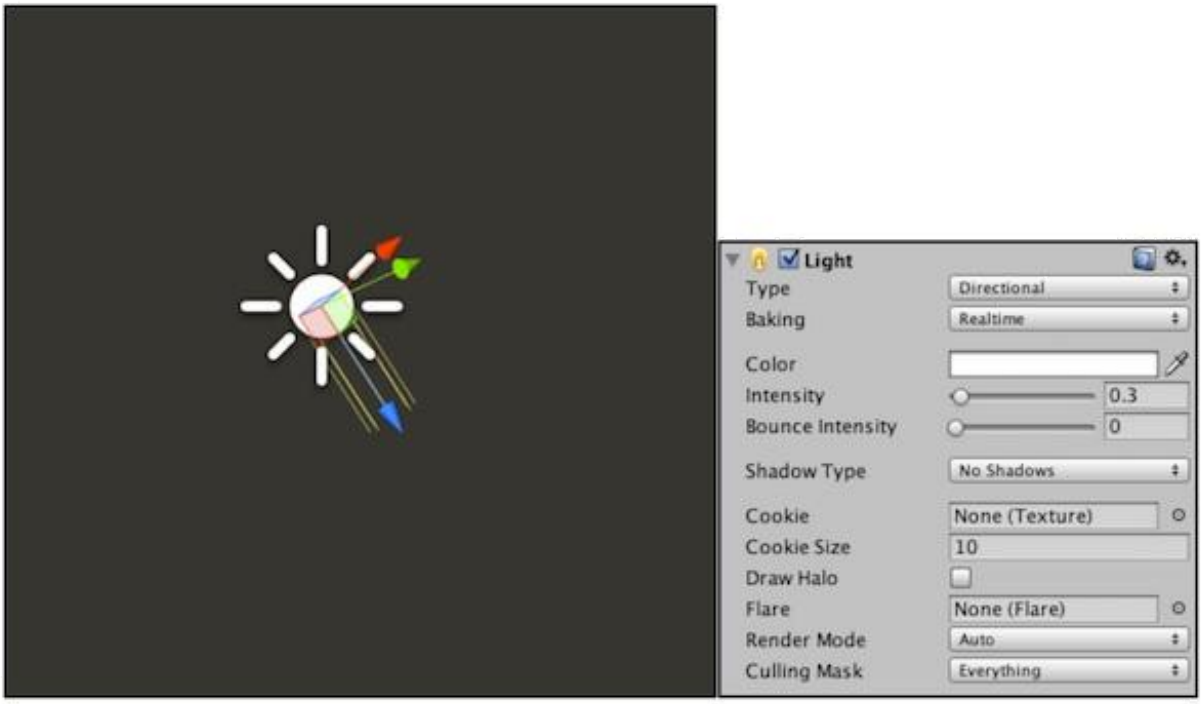

可实现的光照效果：
✓ 光照纹理
✓ 环境光
✓ 自发光
✓ 阴影（平行光的阴影）
Base Pass
渲染设置
Tags { "LightMode"="ForwardBase"}
#pragma multi_compile_fwdbase
光照计算
一个逐像素的平行光以及所有逐顶点和SH光源
可实现的光照效果：
✓ 默认情况下不支持阴影，但可以通过使用#pragma multi_compile_fwdadd_fullshadows编译指令来开启阴影
Additional Pass
渲染设置
Tags { "LightMode"="ForwardAdd"}
Blend One One
#pragma multi_compile_fwdadd
光照计算
其他影响该物体的逐像素光源
每个光源执行一次Pass

Figure 9.4 prior to rendering two Pass species



parallel lightFigure 9.5

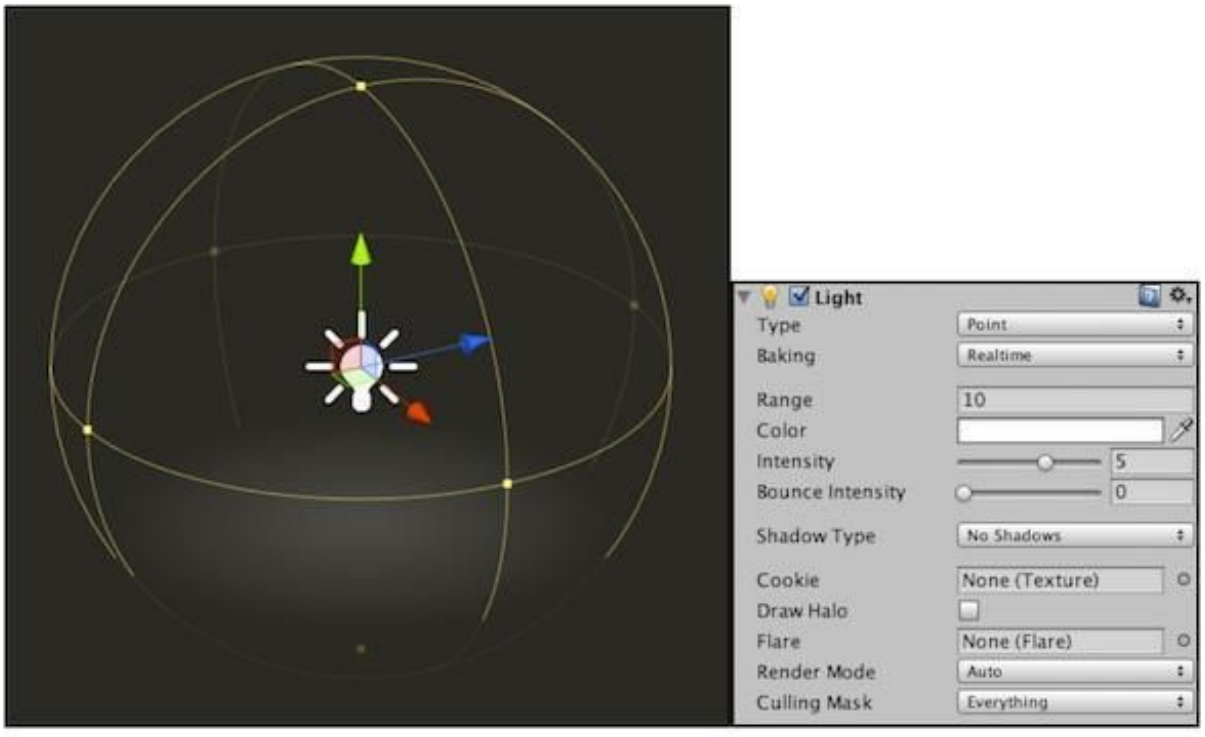


Figure 9.6point light source

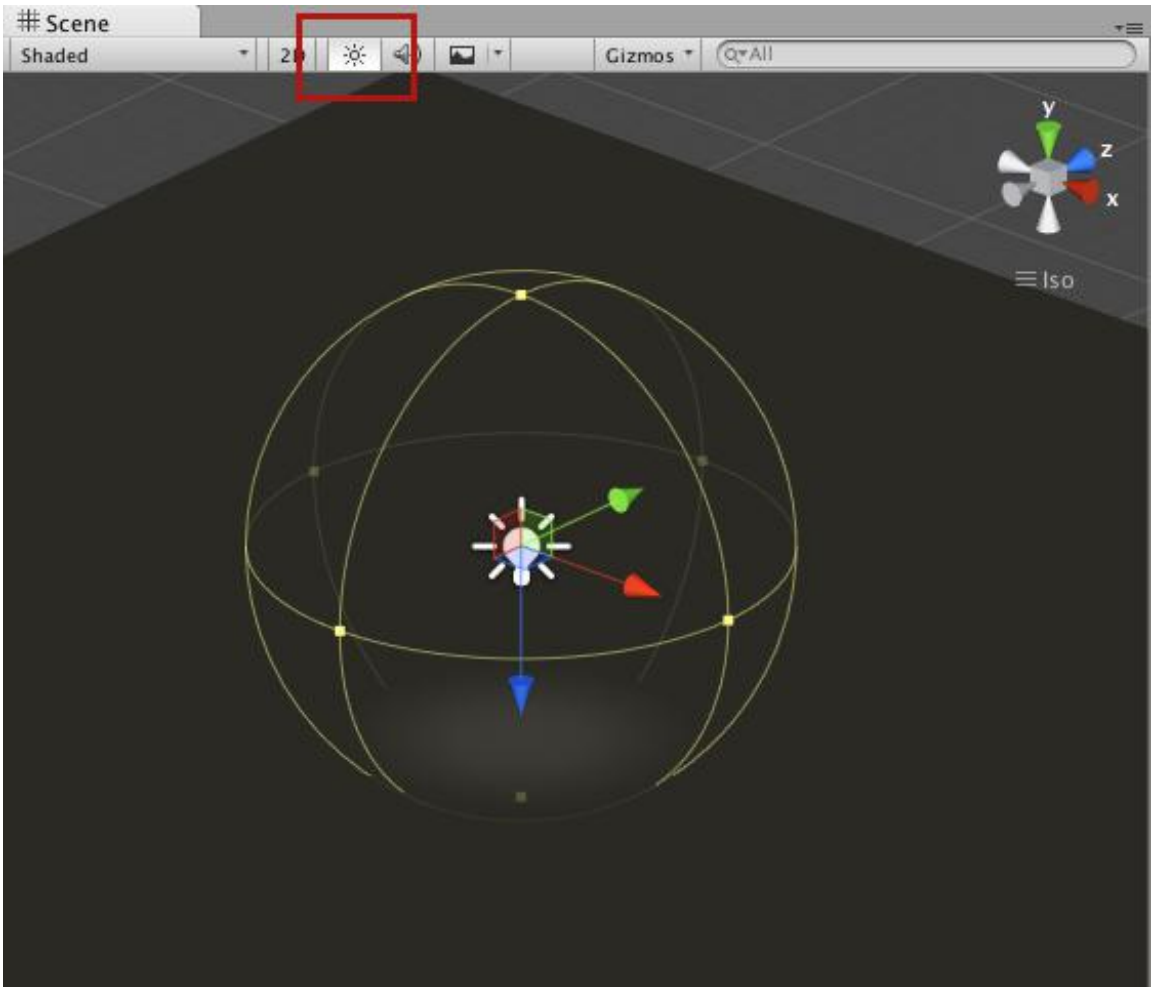
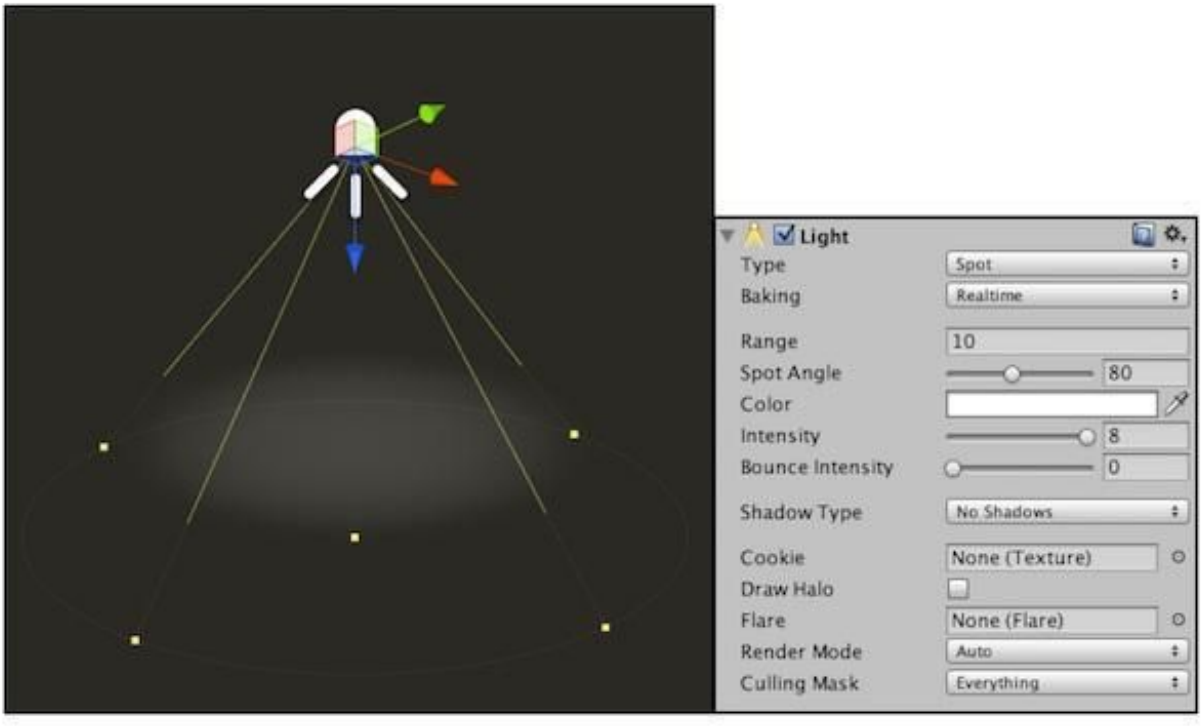
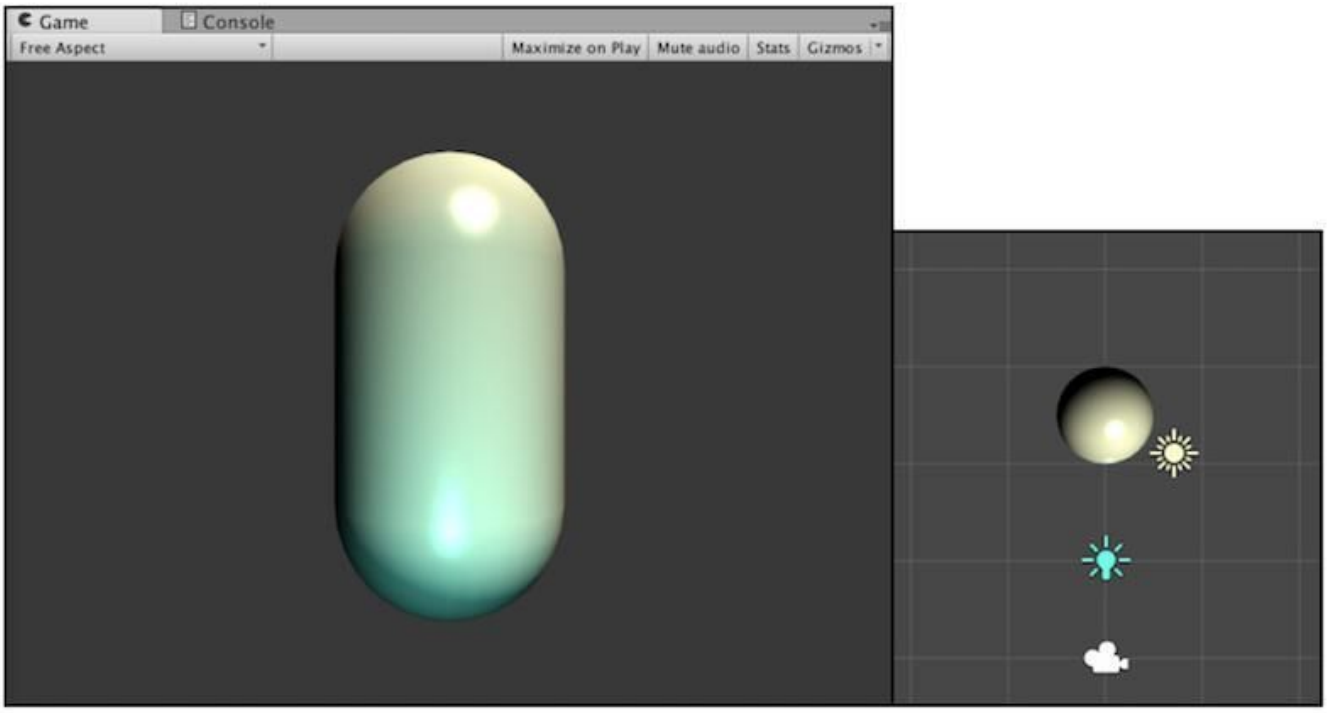


Figure 9.7light is turned onScene view



in FIGspotlight 9.8



using a 9.9 A line light source common point of the object illuminated. The right panel shows the capsule body, the relative position of the point light source is parallel light in the scene of

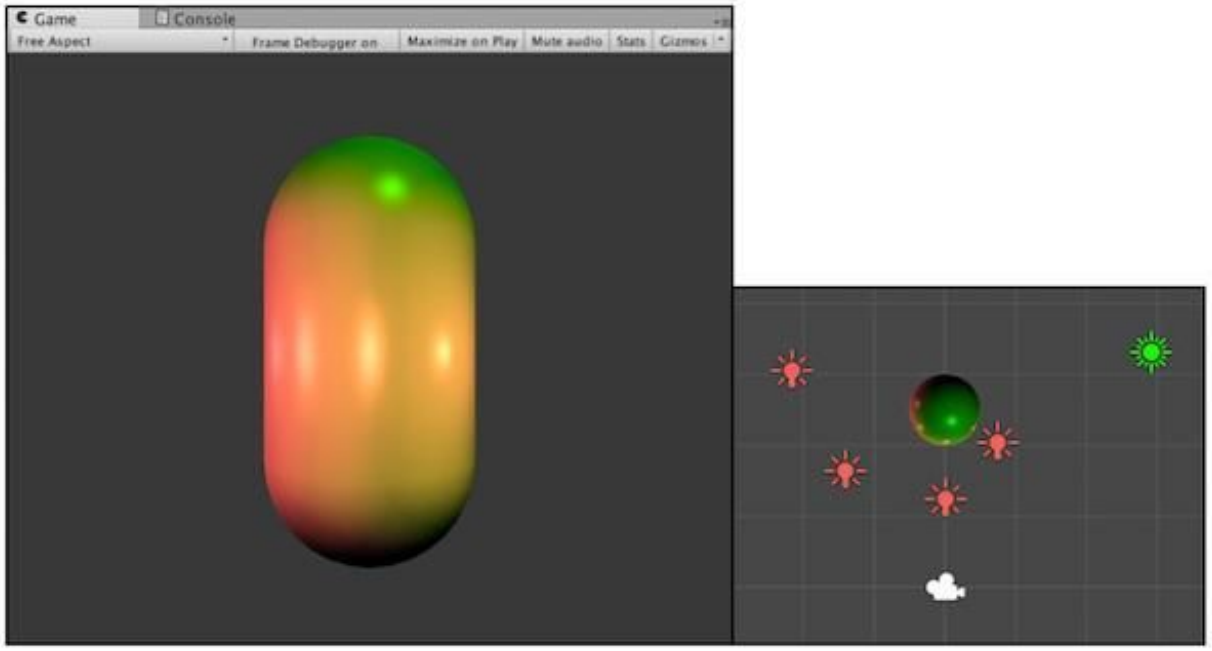
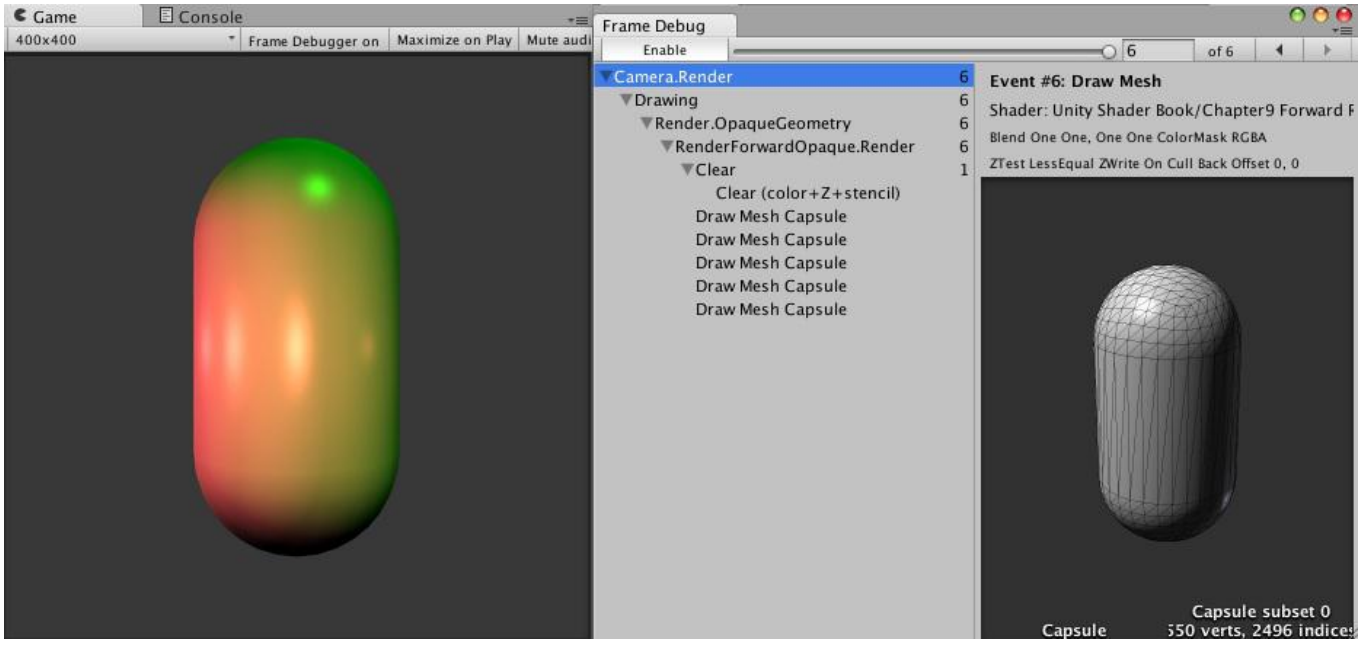
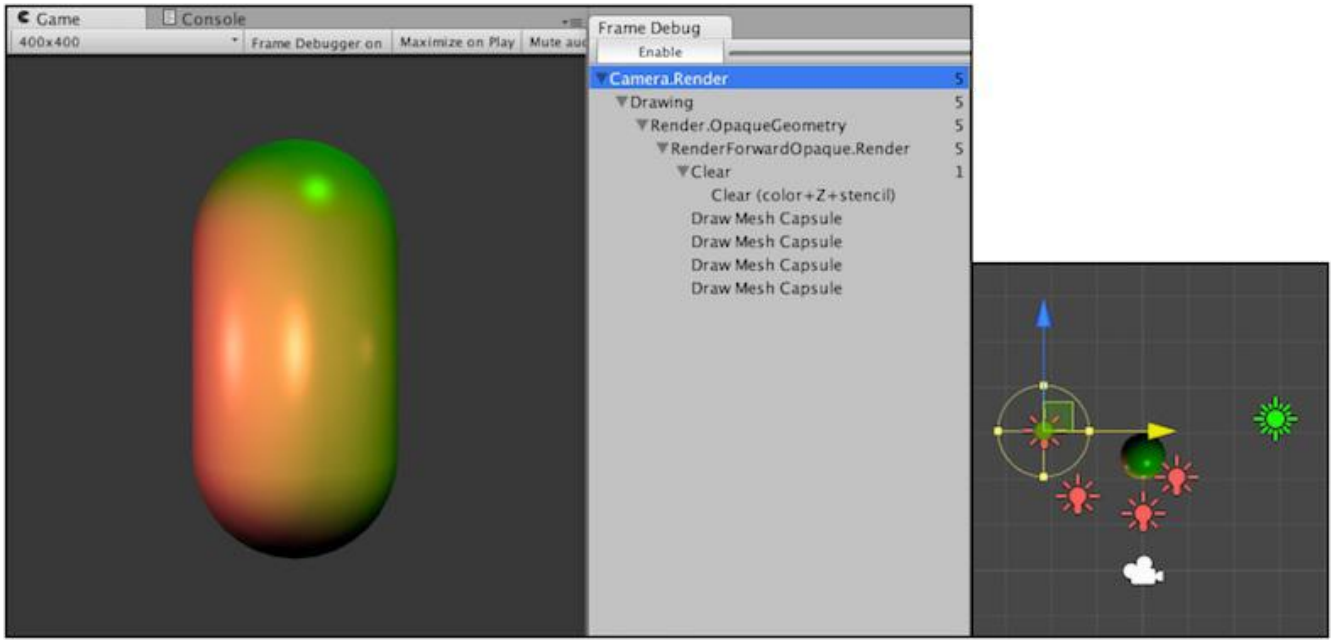


FIG. 9.10 using a 4-point parallel light + light source illuminates an object



9.11 debugger open frame view a scene event are plotted

in FIG. 9.12 cases six events rendering, the rendering sequence is from left to right, top to bottom9.13



in FIG. if the object is not within a range of the illumination source (seen from the right, not the leftmost point light source illuminating the capsule body the range), Unity Additional Pass is not called to processthe light source for the object

view ofRender 9.14 when the light source is set to Mode not Important, the light sources will not be processed by the pixel-wise light

source is turned on in 9.15renderings

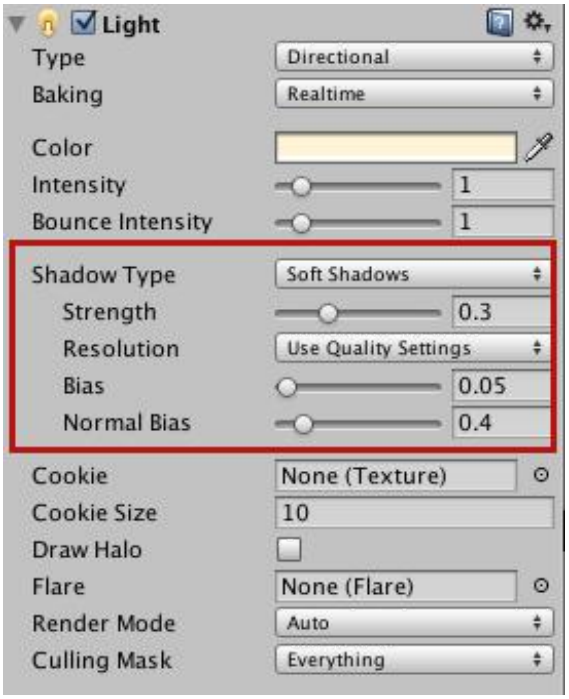


FIG shadow

Cast shadows and Receive shadows properties 9.16 Mesh Renderer module can control whether the object is a projection / receiver shadows

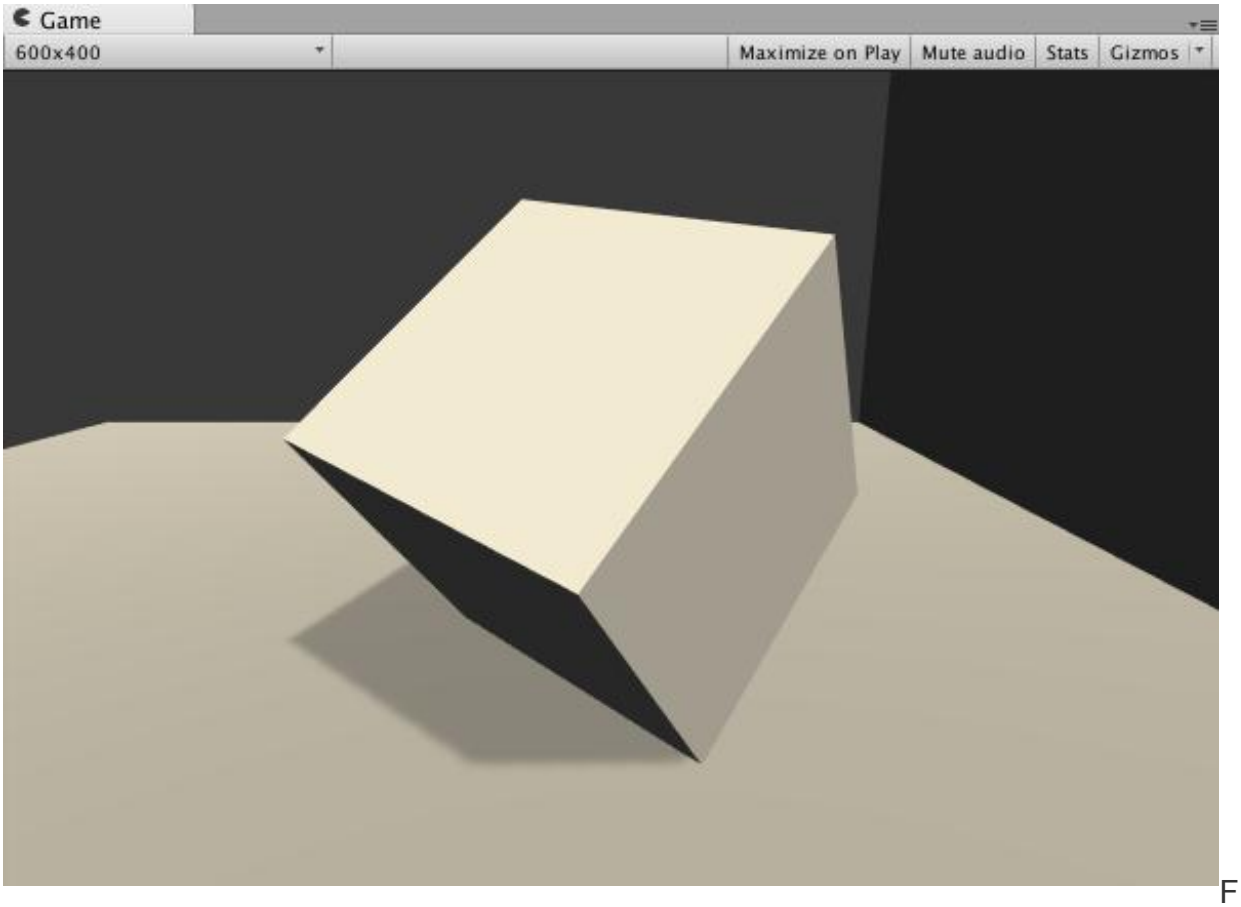
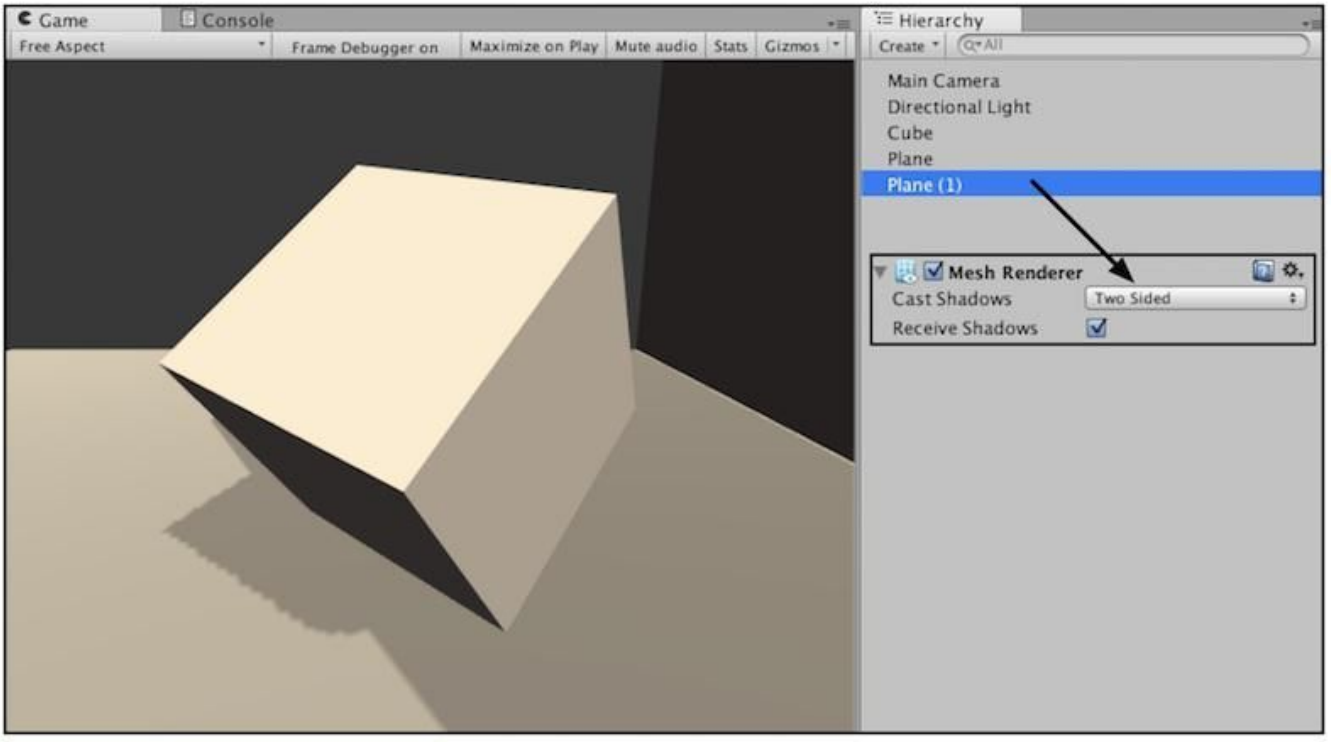
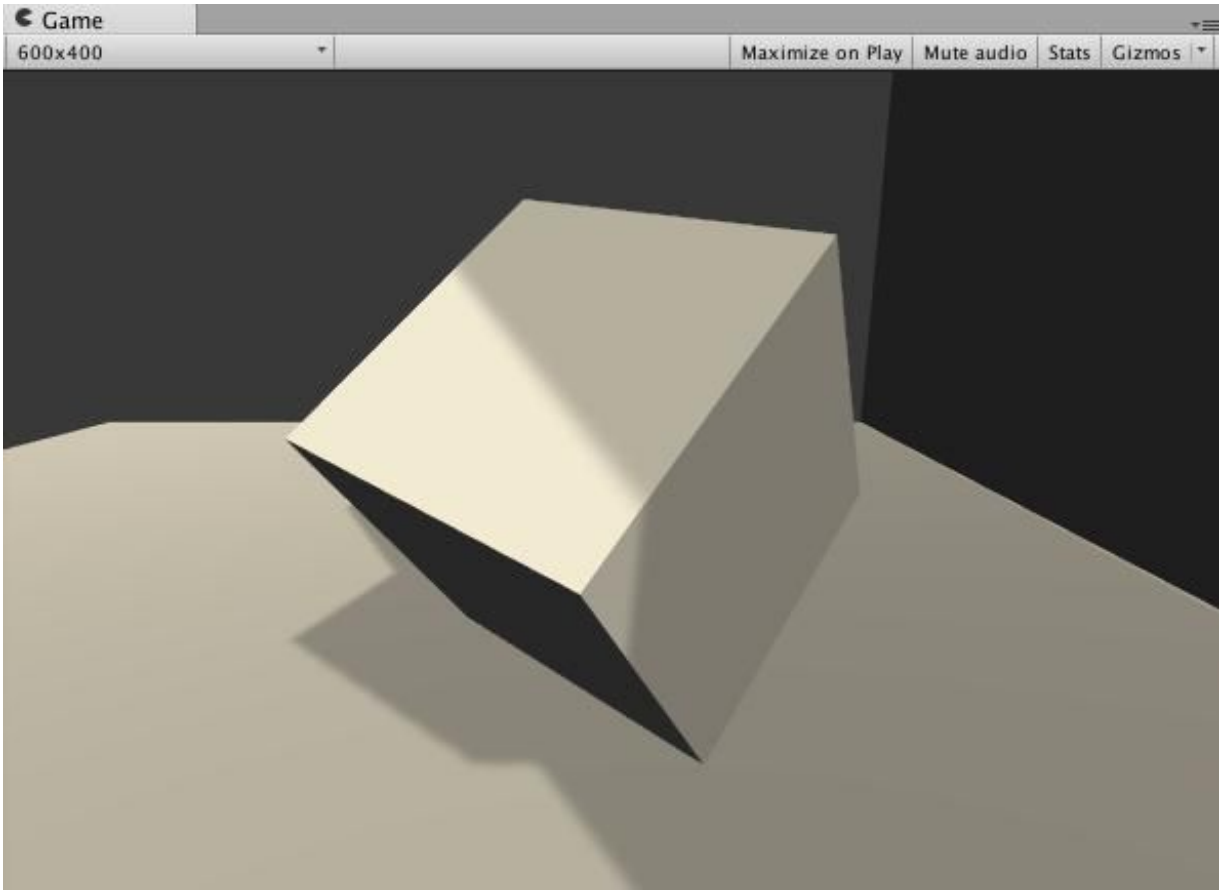


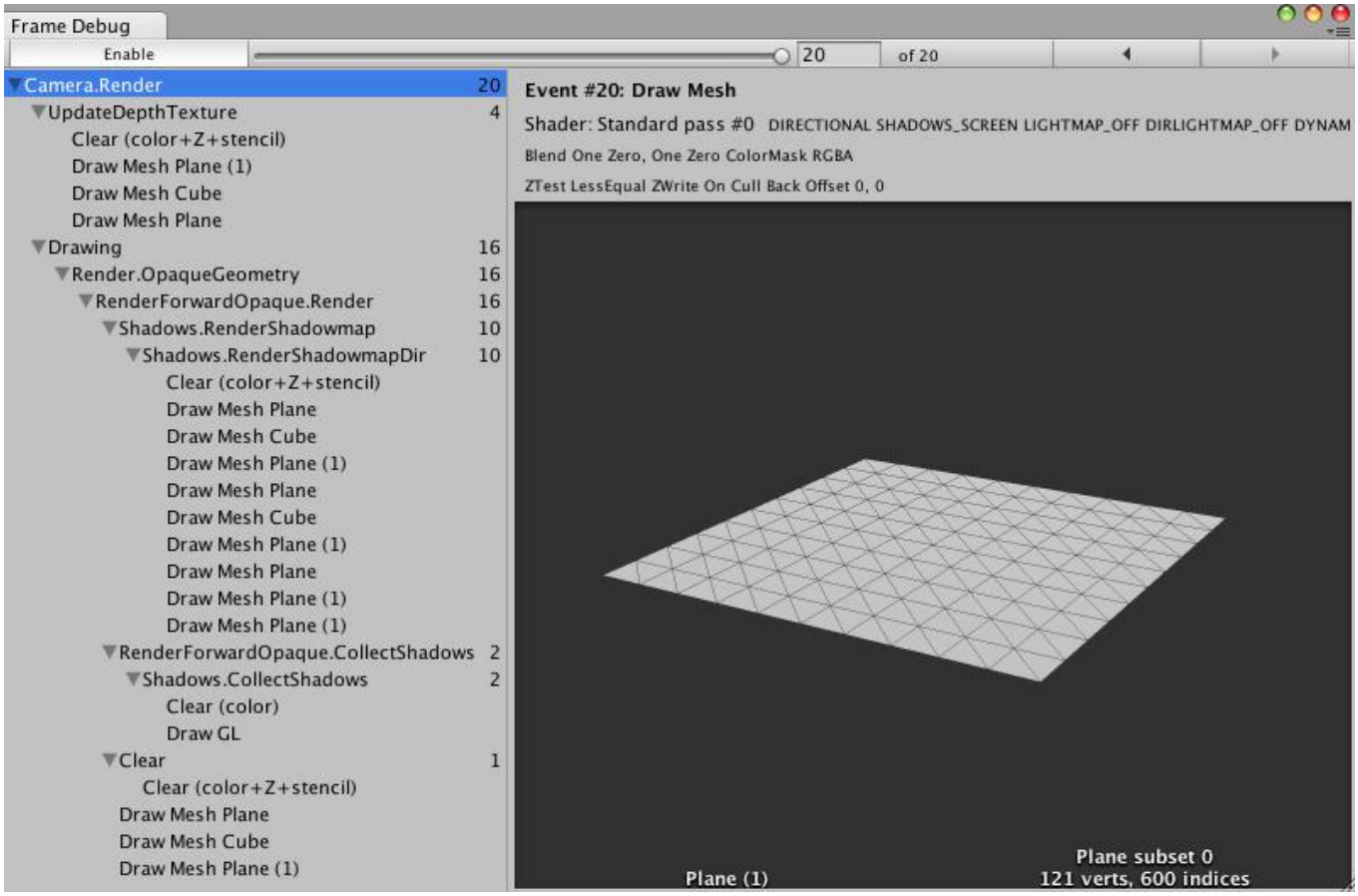
FIG 9.17 open Cast shadows and Receive shadows, so that cube can be cast and receive shadow



map 9.18 Cast shadows set Two Sided lets backlight surface plane of the right shadowalso produces



map9.19 cubic shadowmay be received from the right plane


Frame Debug
Enable
20
of 20
Camera.Render 20
UpdateDepthTexture 4
Clear (color+Z+stencil)
Draw Mesh Plane (1)
Draw Mesh Cube
Draw Mesh Plane
Drawing 16
Render.OpaqueGeometry 16
RenderForwardOpaque.Render 16
Shadows.RenderShadowmap 10
Shadows.RenderShadowmapDir 10
Clear (color+Z+stencil)
Draw Mesh Plane
Draw Mesh Cube
Draw Mesh Plane (1)
Draw Mesh Plane
Draw Mesh Cube
Draw Mesh Plane (1)
Draw Mesh Plane
Draw Mesh Plane (1)
Draw Mesh Plane (1)
RenderForwardOpaque.CollectShadows 2
Shadows.CollectShadows 2
Clear (color)
Draw GL
Clear 1
Clear (color+Z+stencil)
Draw Mesh Plane
Draw Mesh Cube
Draw Mesh Plane (1)
Event #20: Draw Mesh
Shader: Standard pass #0 DIRECTIONAL SHADOWS_SCREEN LIGHTMAP_OFF DIRLIGHTMAP_OFF DYNAM
Blend One Zero, One Zero ColorMask RGBA
ZTest LessEqual ZWrite On Cull Back Offset 0, 0
Plane (1)
Plane subset 0
121 verts, 600 indices

mapframe using 9.20 debugging View the shadow rendering process of

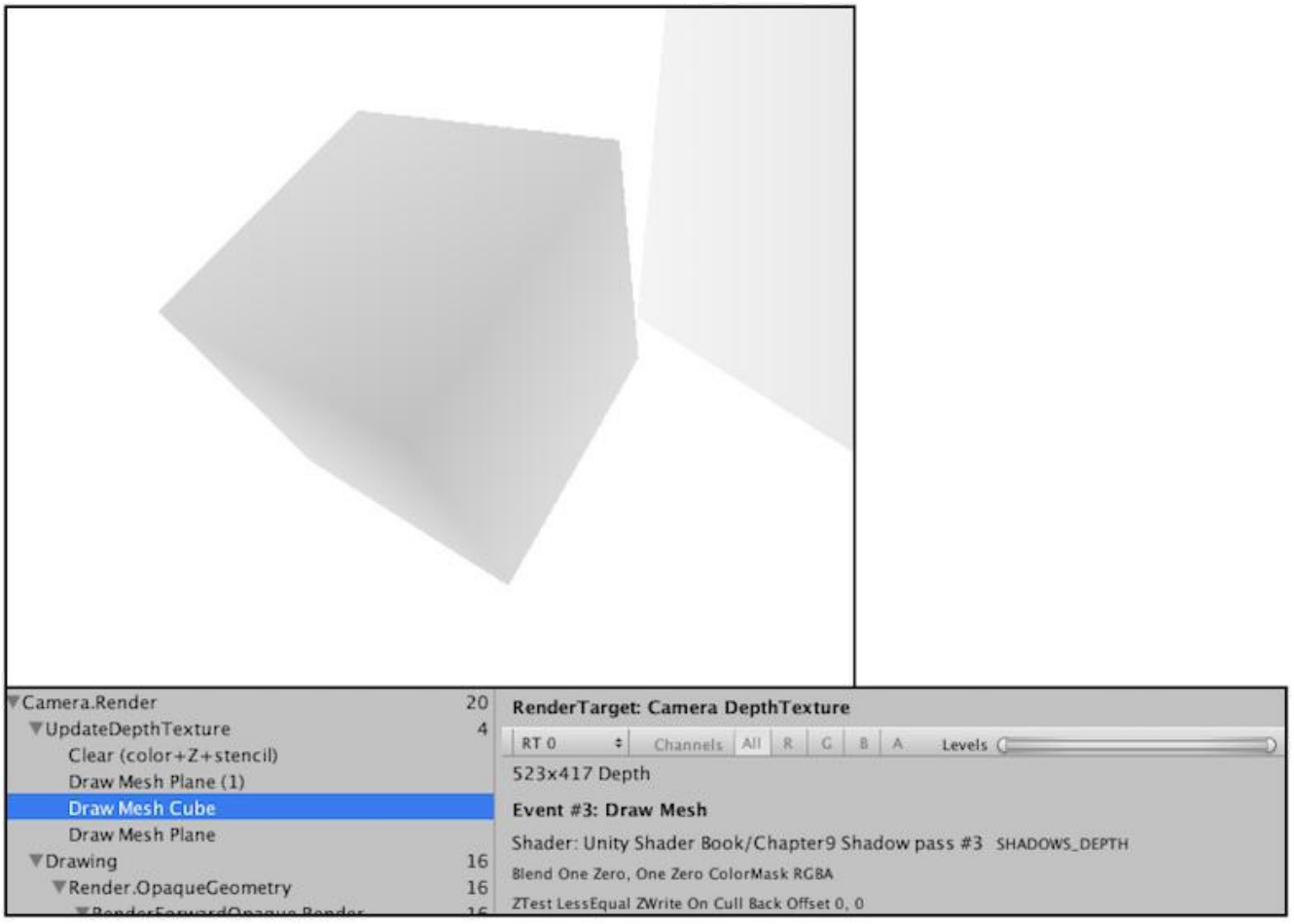
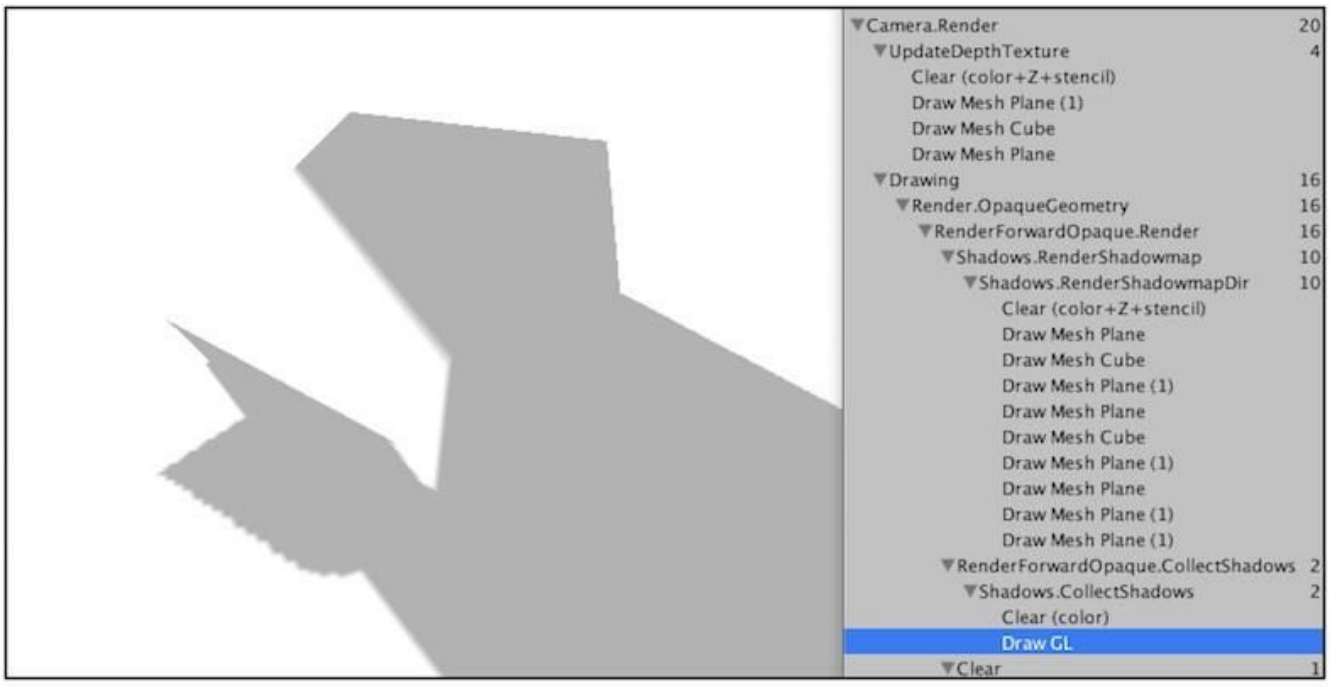
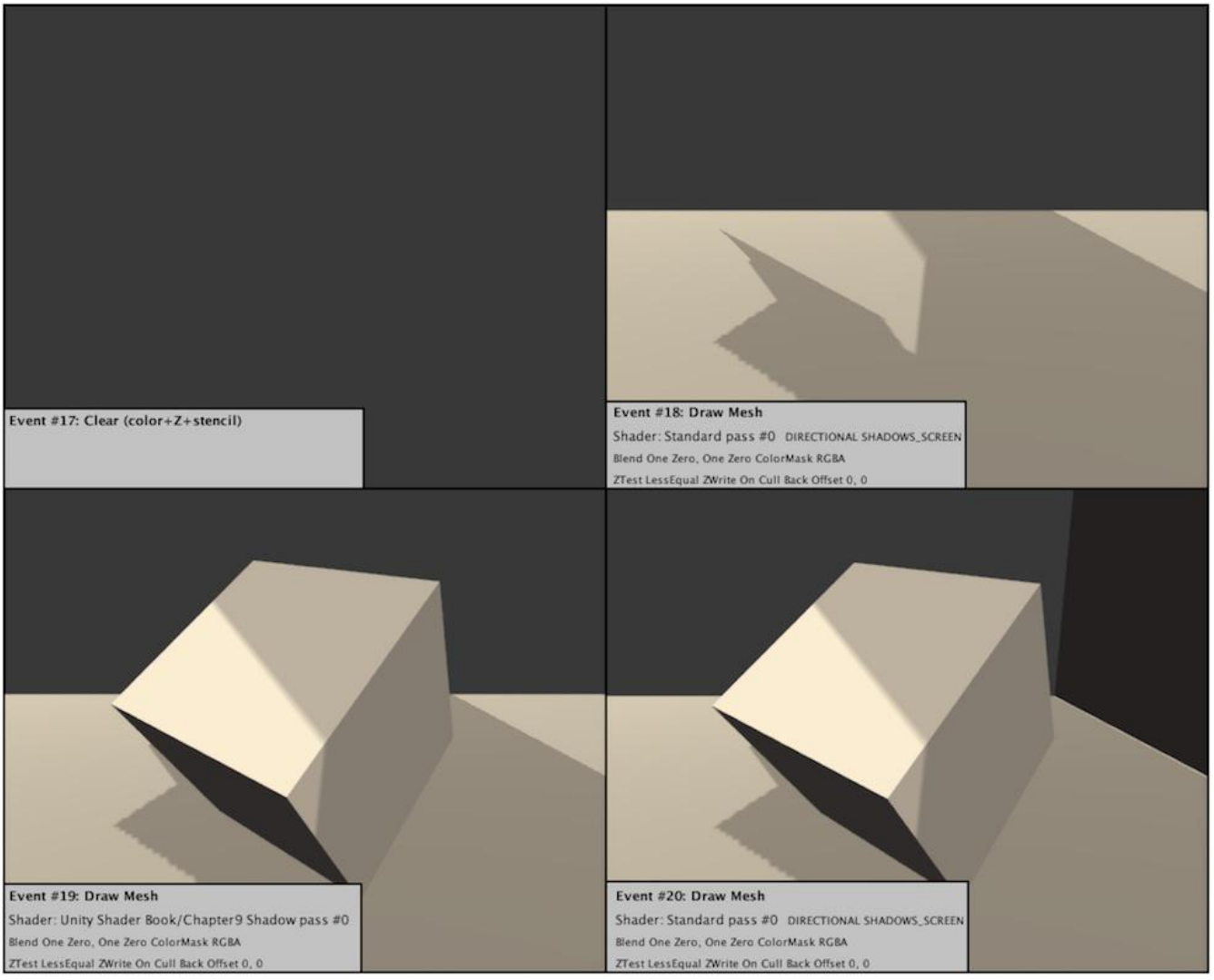

Camera.Render 20
UpdateDepthTexture 4
Clear (color+Z+stencil)
Draw Mesh Plane (1)
Draw Mesh Cube
Draw Mesh Plane
Drawing 16
Render.OpaqueGeometry 16
RenderTarget: Camera DepthTexture
RT 0
Channels All R G B A Levels
523x417 Depth
Event #3: Draw Mesh
Shader: Unity Shader Book/Chapter9 Shadow pass #3 SHADOWS_DEPTH
Blend One Zero, One Zero ColorMask RGBA
ZTest LessEqual ZWrite On Cull Back Offset 0, 0

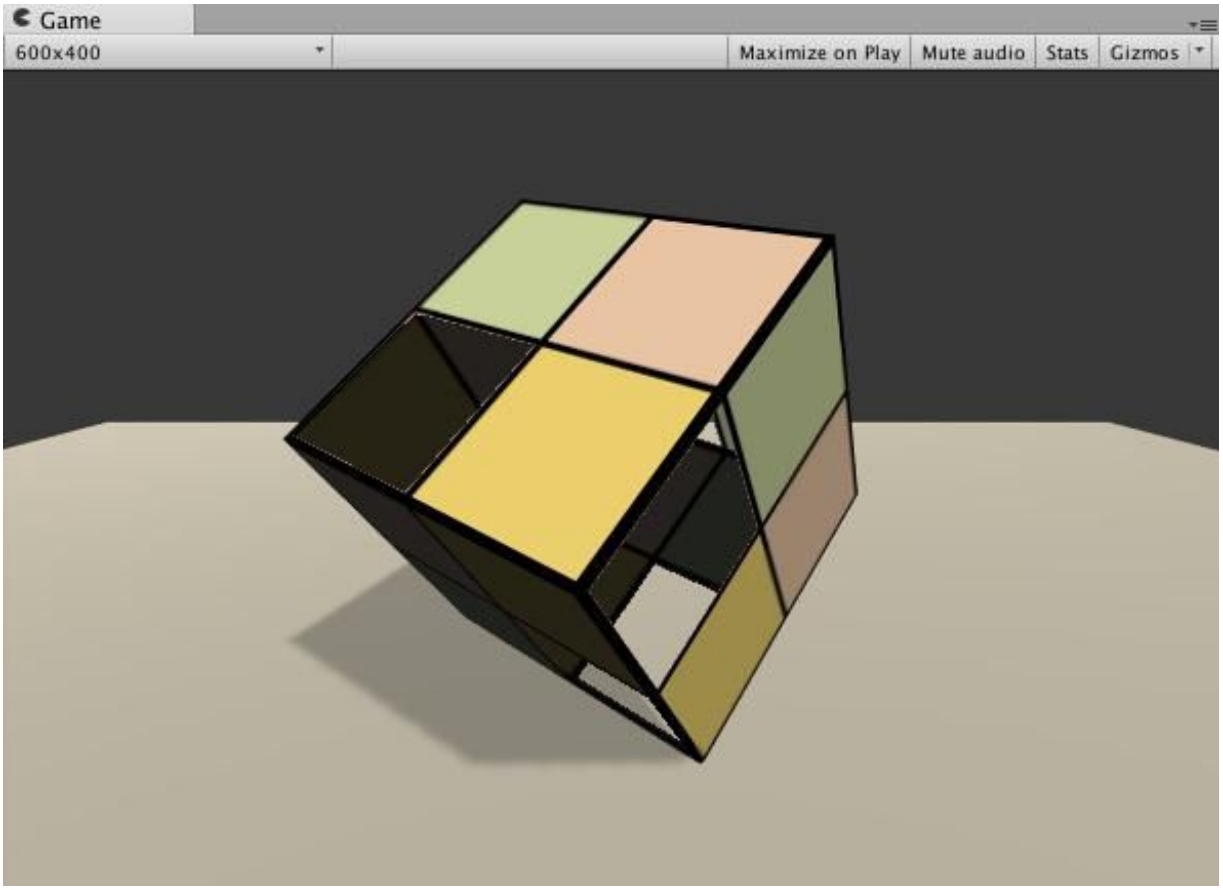
FIG9.21 cubicupdateresult depth of the texture



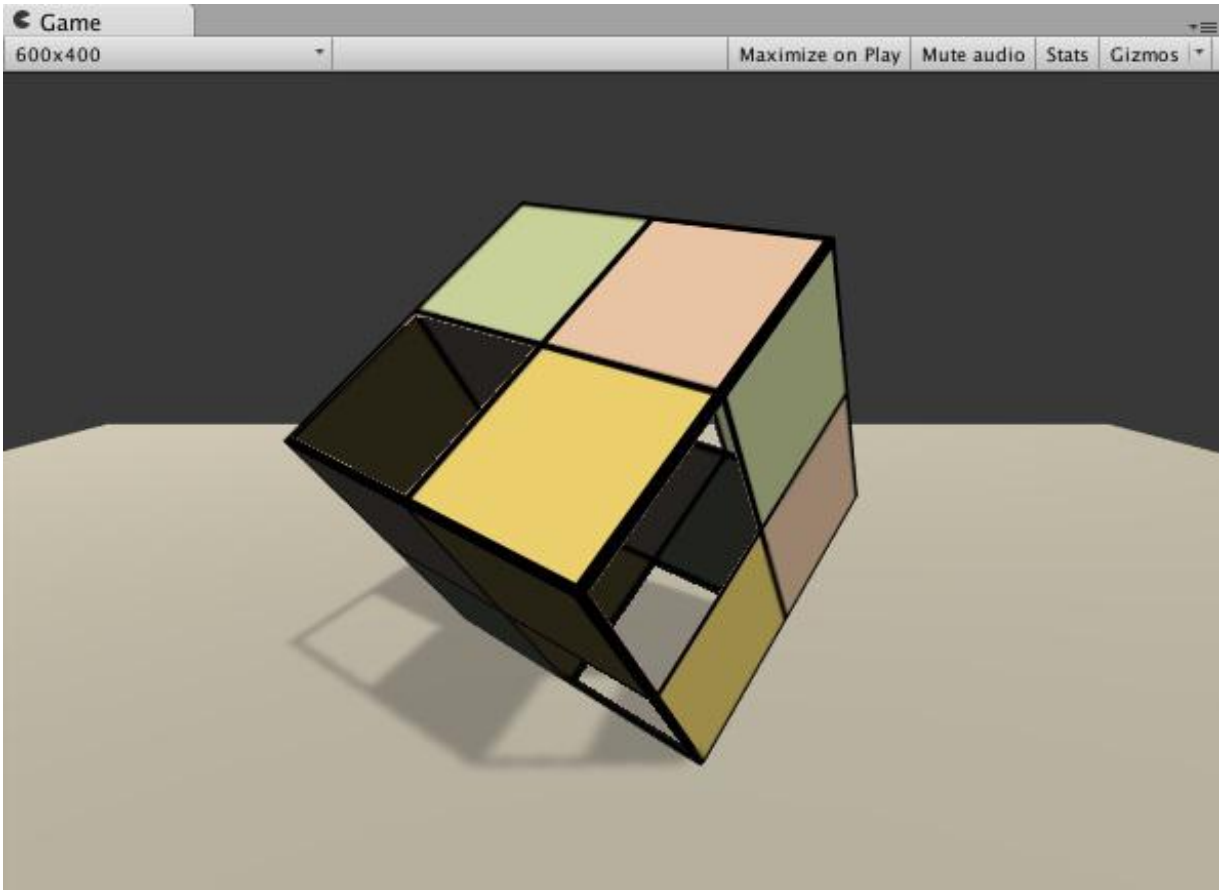
shaded 9.22 screen space versus


Event #17: Clear (color+Z+stencil)
Event #18: Draw Mesh
Shader: Standard pass #0 DIRECTIONAL SHADOWS_SCREEN
Blend One Zero, One Zero ColorMask RGBA
ZTest LessEqual ZWrite On Cull Back Offset 0, 0
Event #19: Draw Mesh
Shader: Unity Shader Book/Chapter9 Shadow pass #0
Blend One Zero, One Zero ColorMask RGBA
ZTest LessEqual ZWrite On Cull Back Offset 0, 0
Event #20: Draw Mesh
Shader: Standard pass #0 DIRECTIONAL SHADOWS_SCREEN
Blend One Zero, One Zero ColorMask RGBA
ZTest LessEqual ZWrite On Cull Back Offset 0, 0

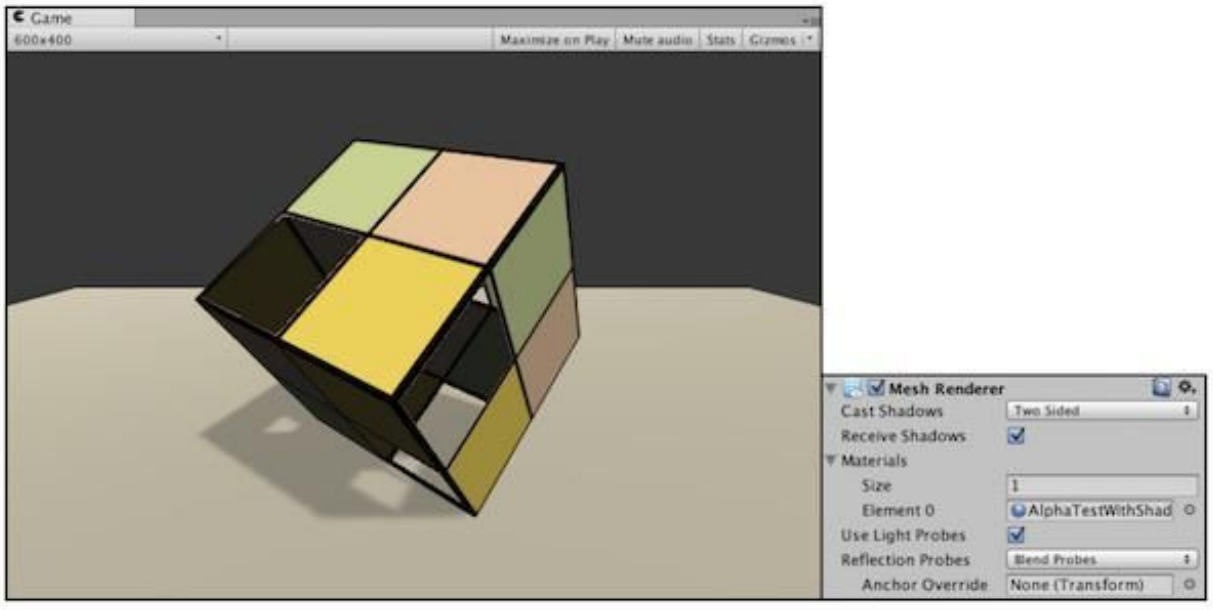
view9.23 Unity process screen shadow



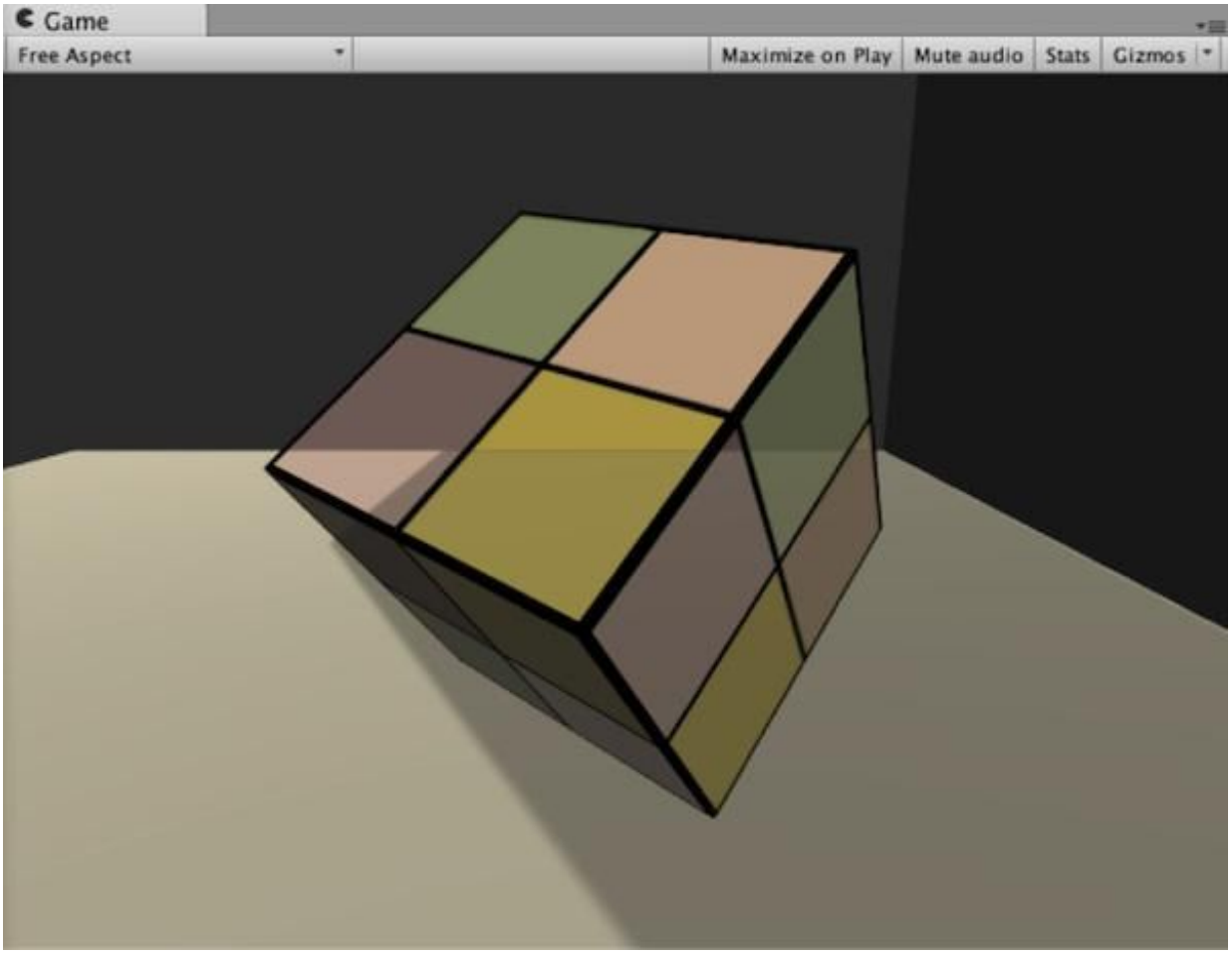
Transparency Testdrawing9.24 can cast shadows of the object



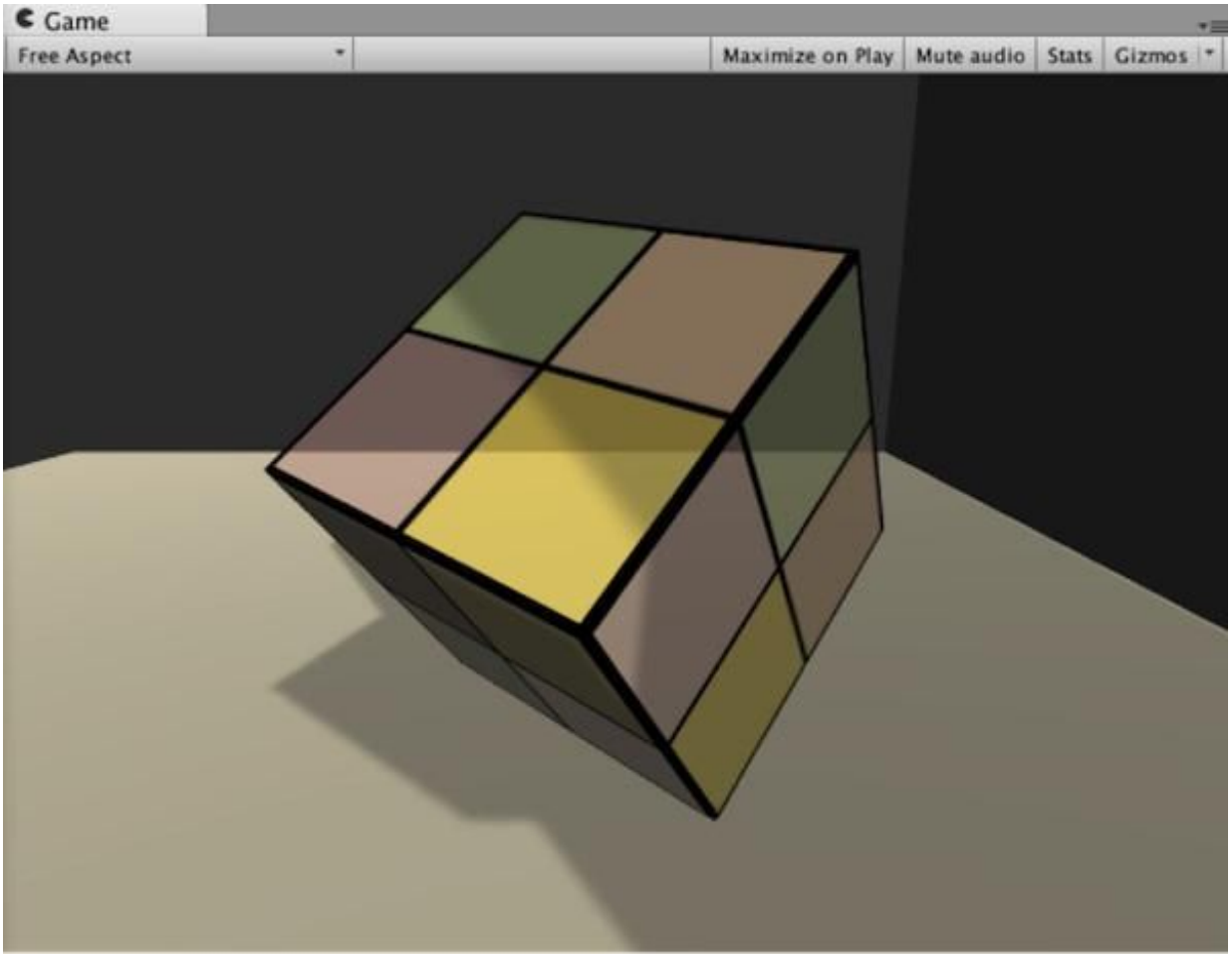
of FIG. 9.25 correctly set Fallback using transparency testobjects



FIGset correctly 9.26 transparency test object property Cast Shadow

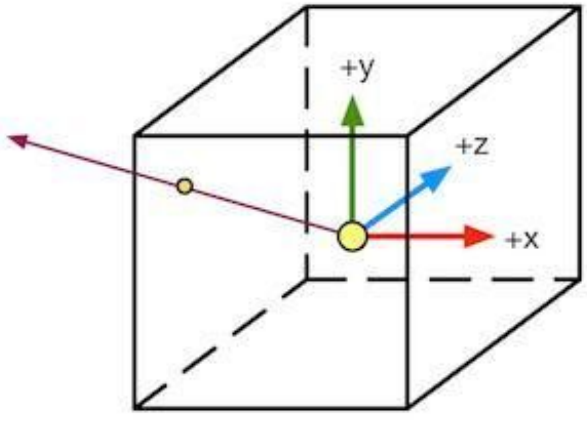


9.27 using FIG transparency mixed Unity Shader Fallback set of built-in Transparent / VertexLit. Translucent objects not cast shadows below the plane, the shadow is not received from the right plane, it looks like the same ascompletely transparent



FIGFallback 9.28VertexLit set to force the generation of the shadow of the object is translucent

Chapter 10Premium textured



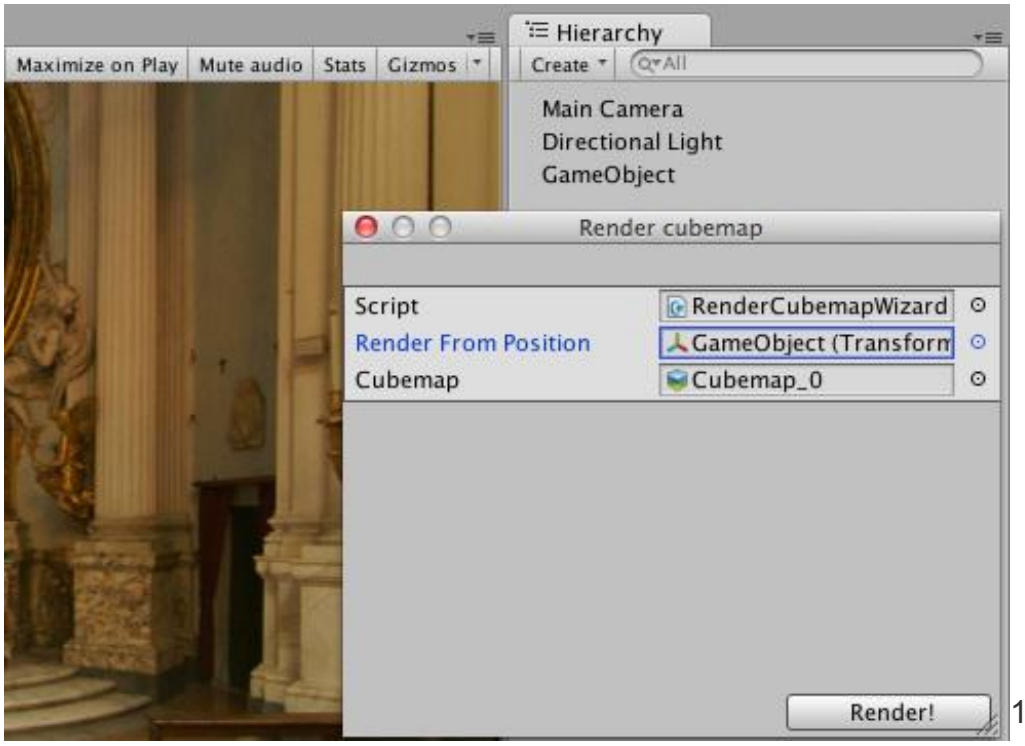
10.1 cube texture sampling

Figure 10.2 sky box material

Figure 10.3 for the sky boxdefined in 

scene to use from the

diagram10.4 using a sky box scenario

Figure 

10.5 using a script to create a cube texture

map 10.6 using a script rendering cube texture

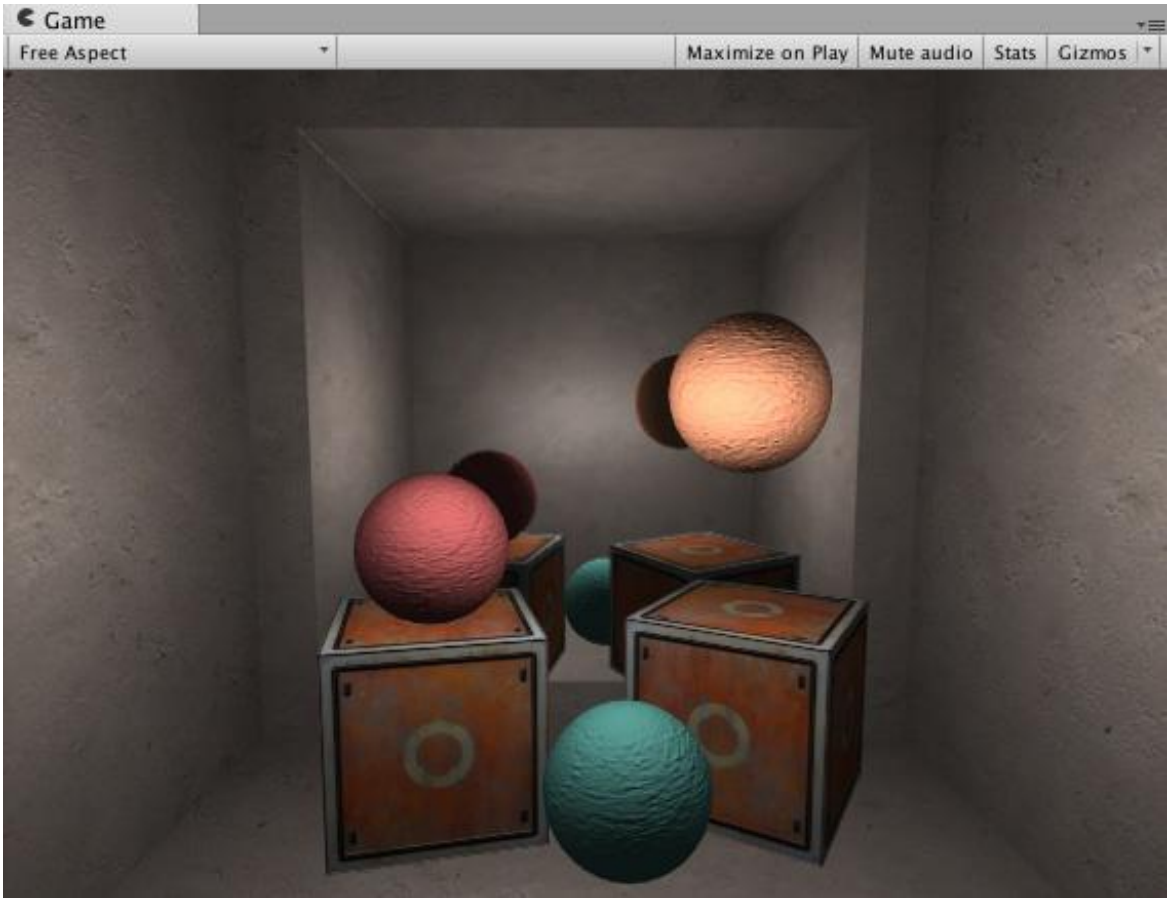
maps

10.7 usingreflection effect Teapot model

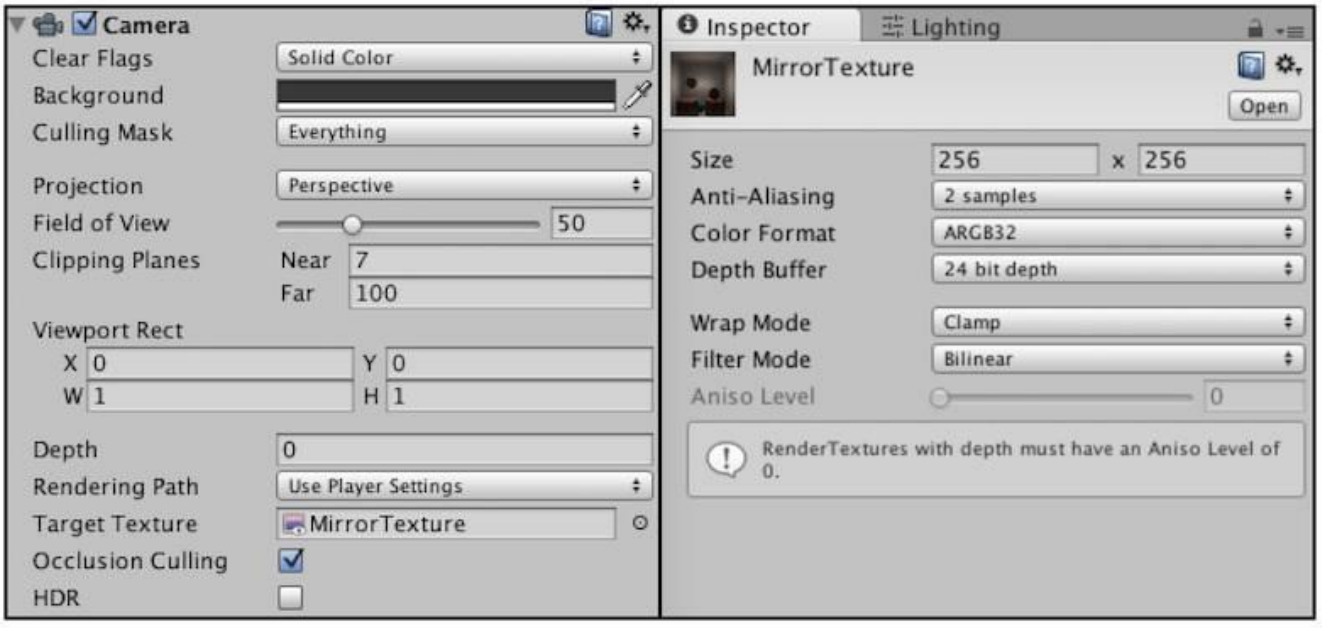
FIG10.8Snell's law

10.9 FIG usingof refraction effects Teapotmodel

FIG 10.10using a Fresnel reflectionTeapot model



mirror FIG 10.11



renderings10.12 left: the camera is set to Target Texture custom render textures.
Right:render texture textures provided

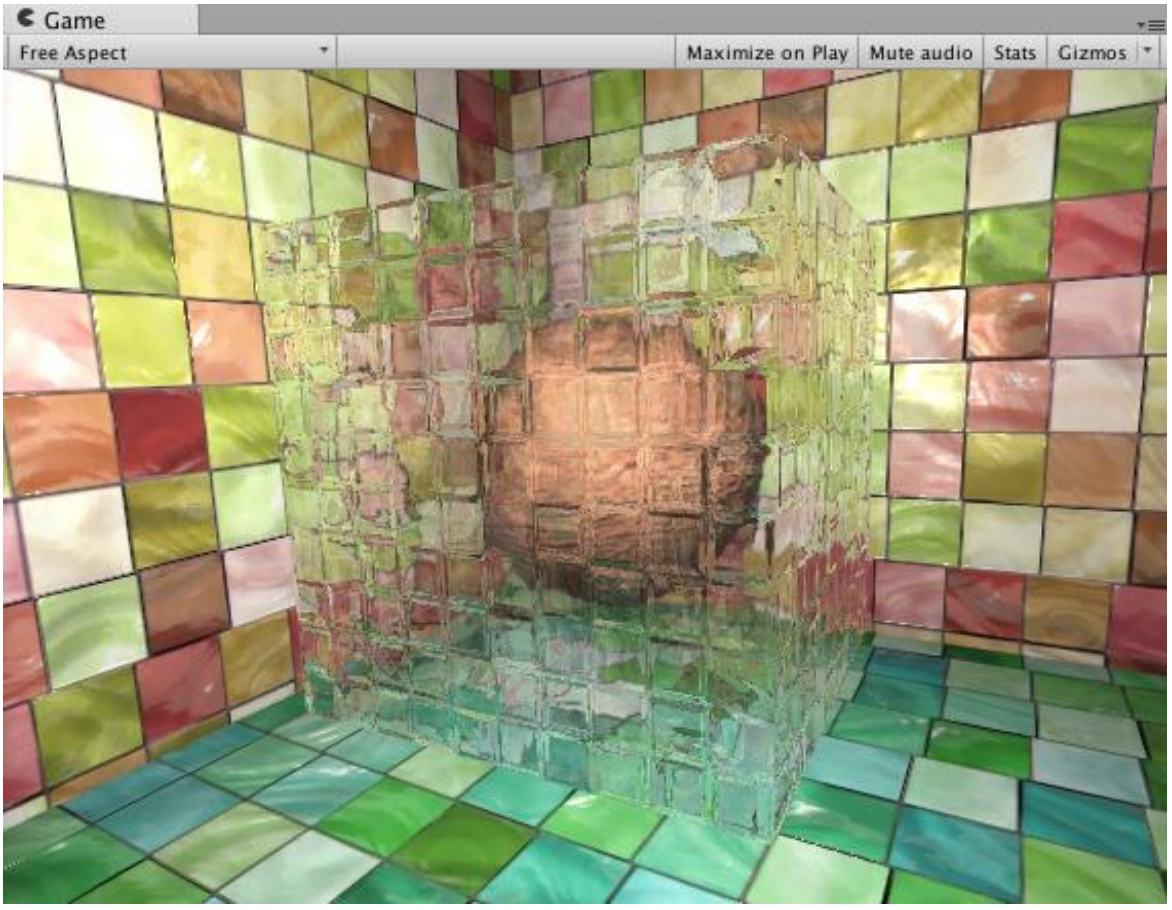
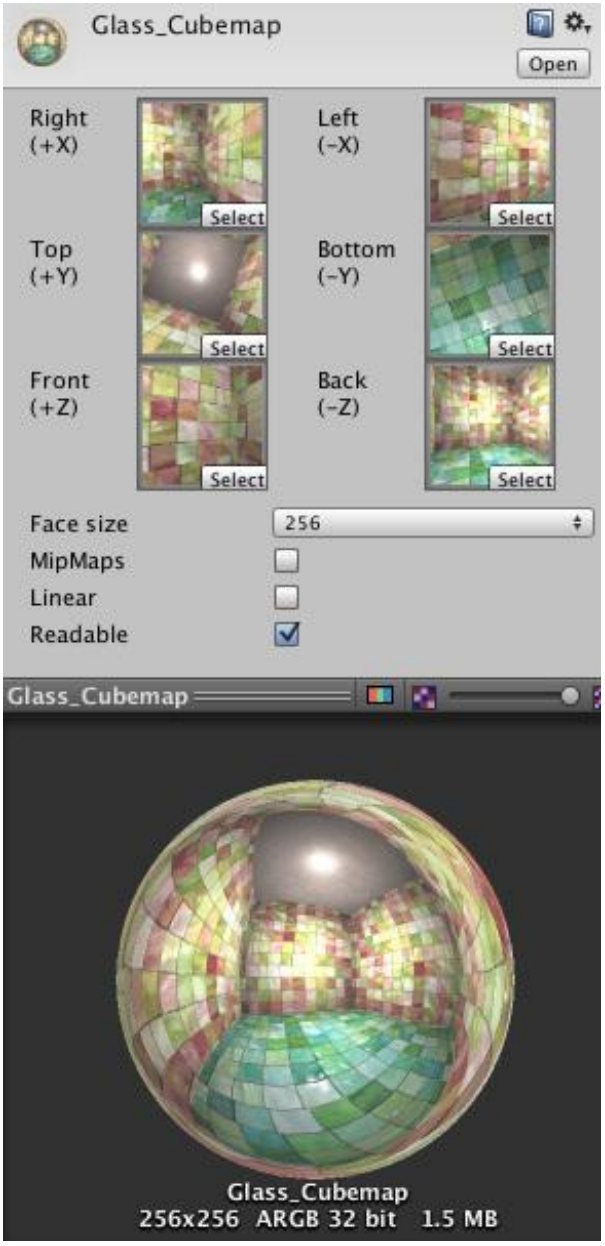
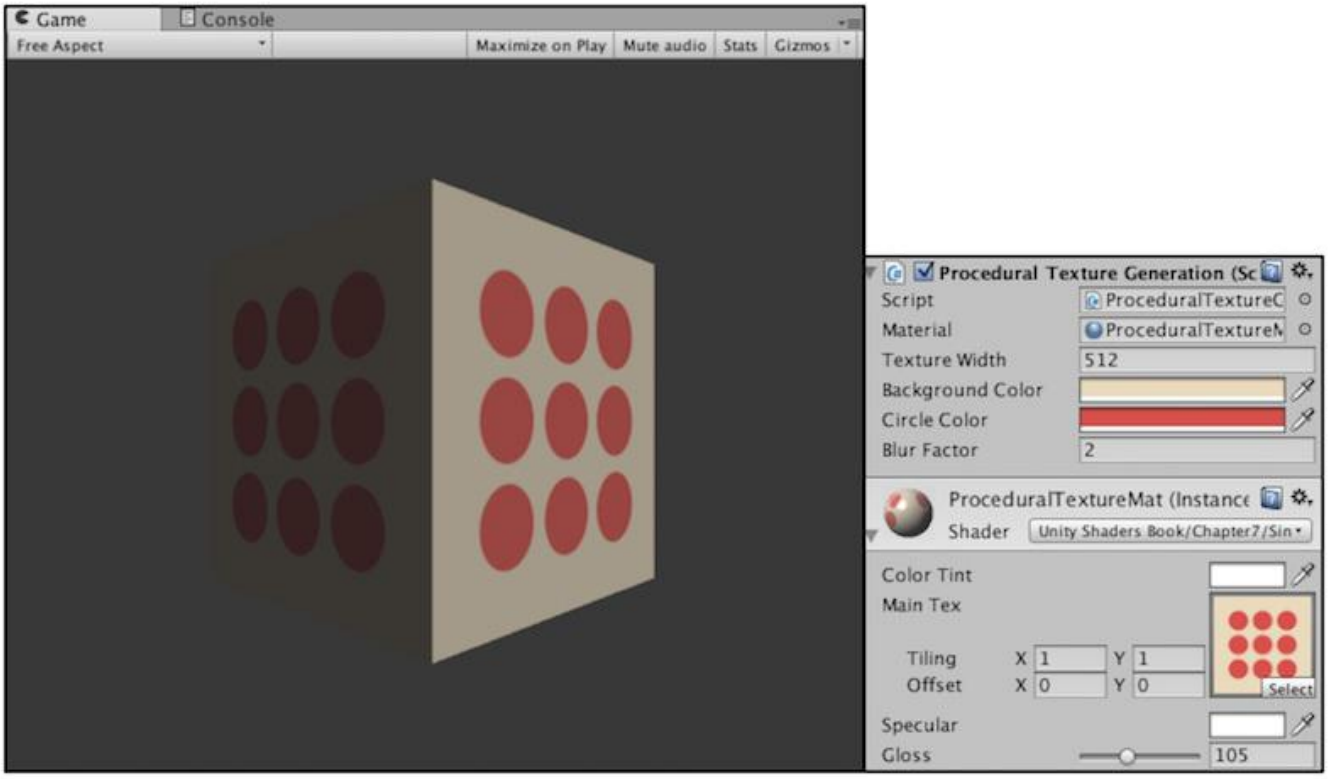


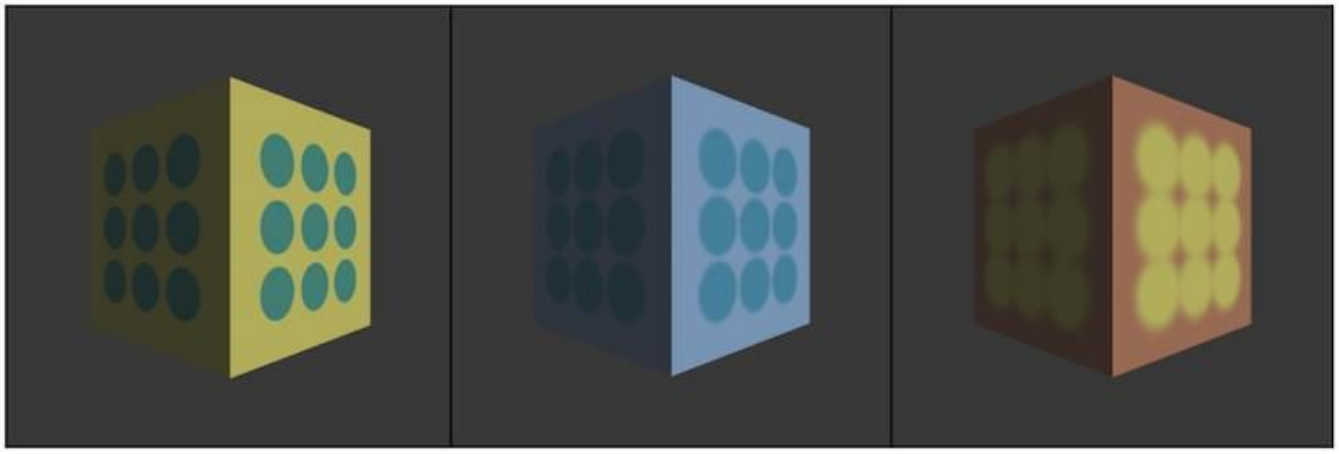
FIG10.13renderingsglass



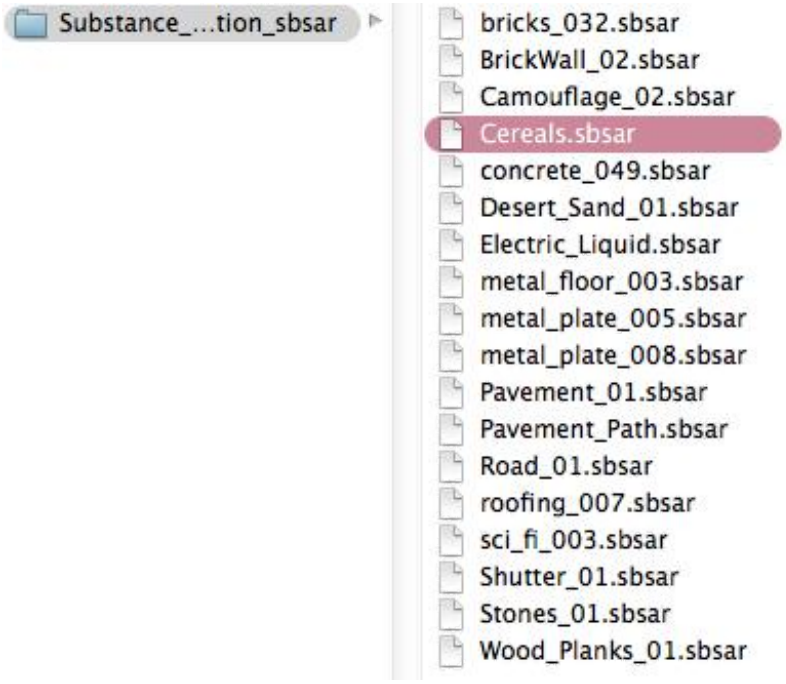
10.14 cube textureused in Example



mapsscript-generated 10.15procedural textures



10.16FIG parameter adjusting procedure to obtain the texture of the different procedural textures



of FIG 10.17 suffix .sbsar material Substance

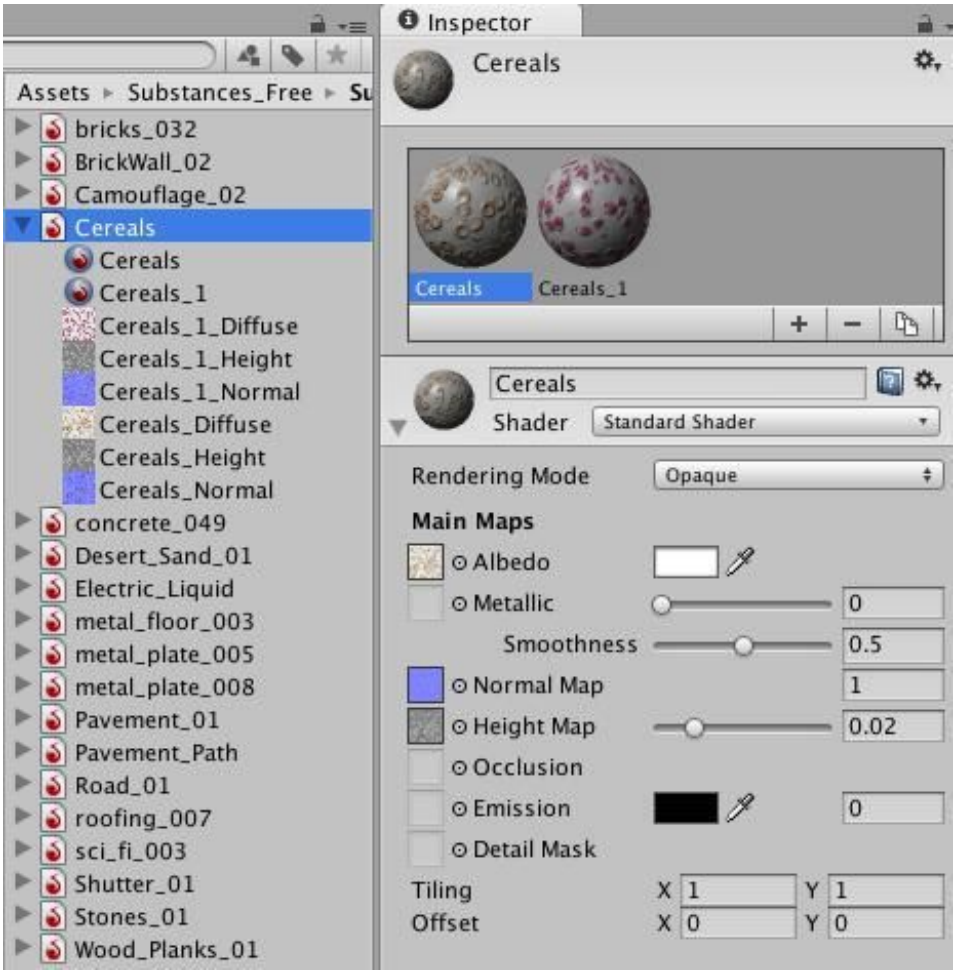


Figure 10.18procedural textures resource

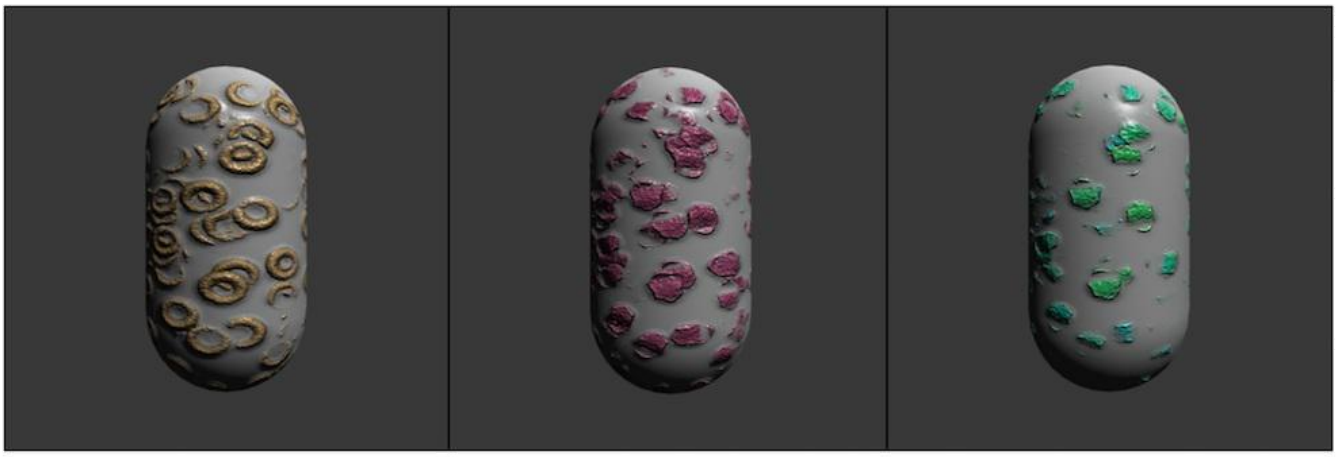


Figure 10.19adjustment program texture attributes can be seemingly completely different program material effects

Chapter 11 allows the screen to move

the frame image sequence Figure 11.111.2section uses

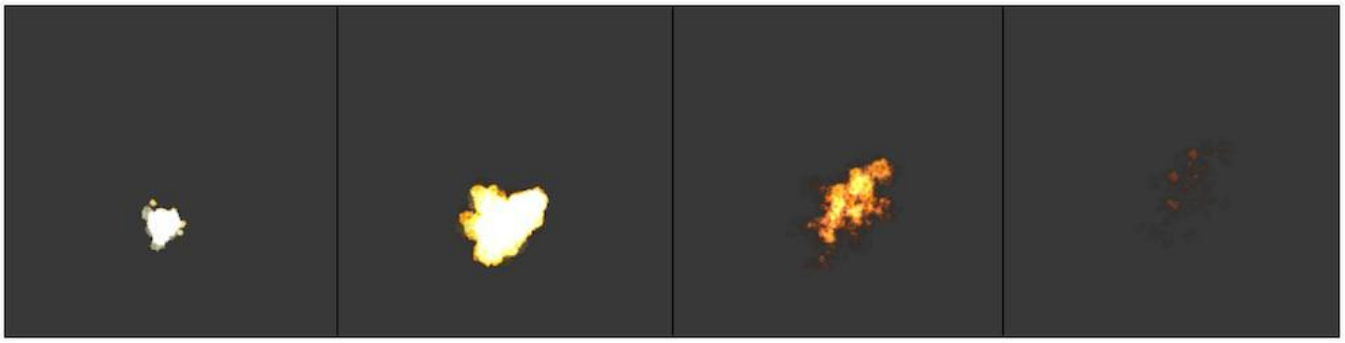
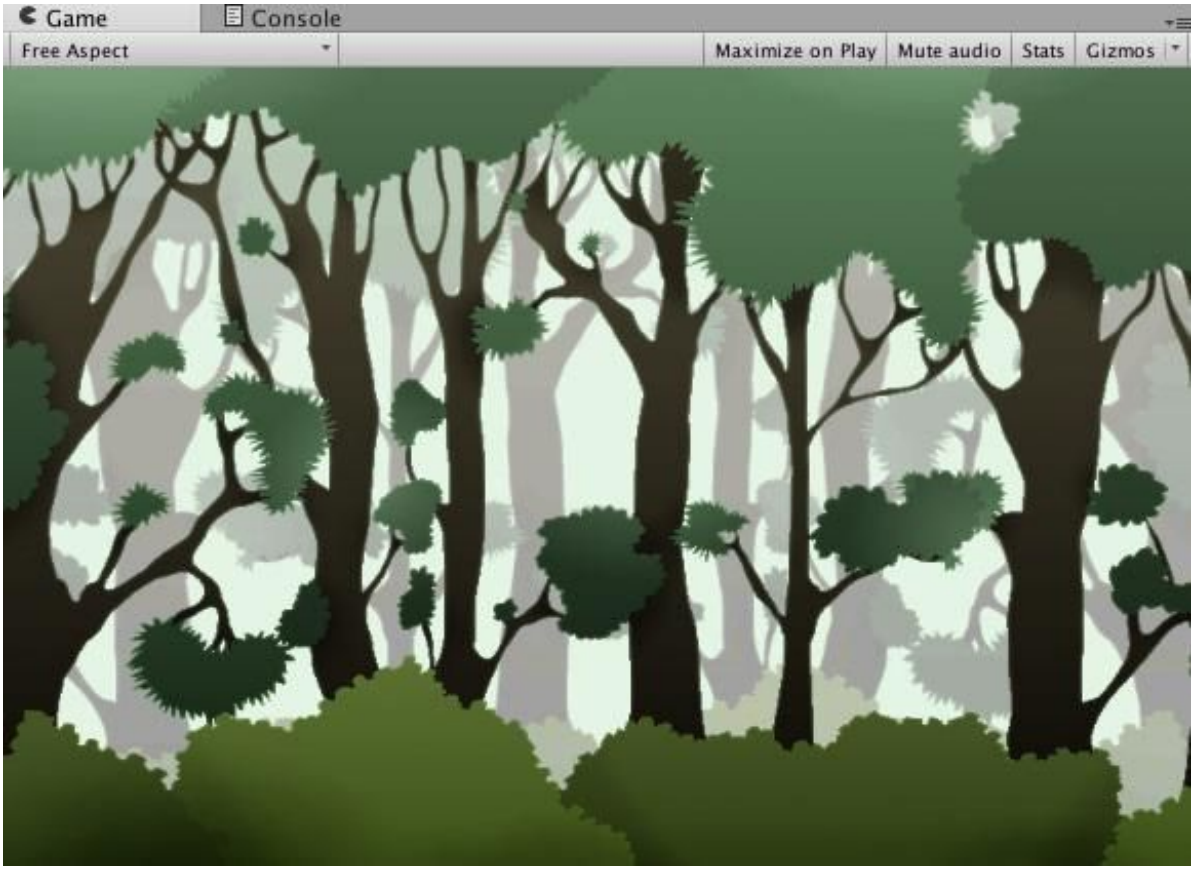


Figure usinganimation sequences frame to implementexplosion effect



11.3 infinitescrolling background (texture source: forest-background © 2012-2013 Julien Jorge

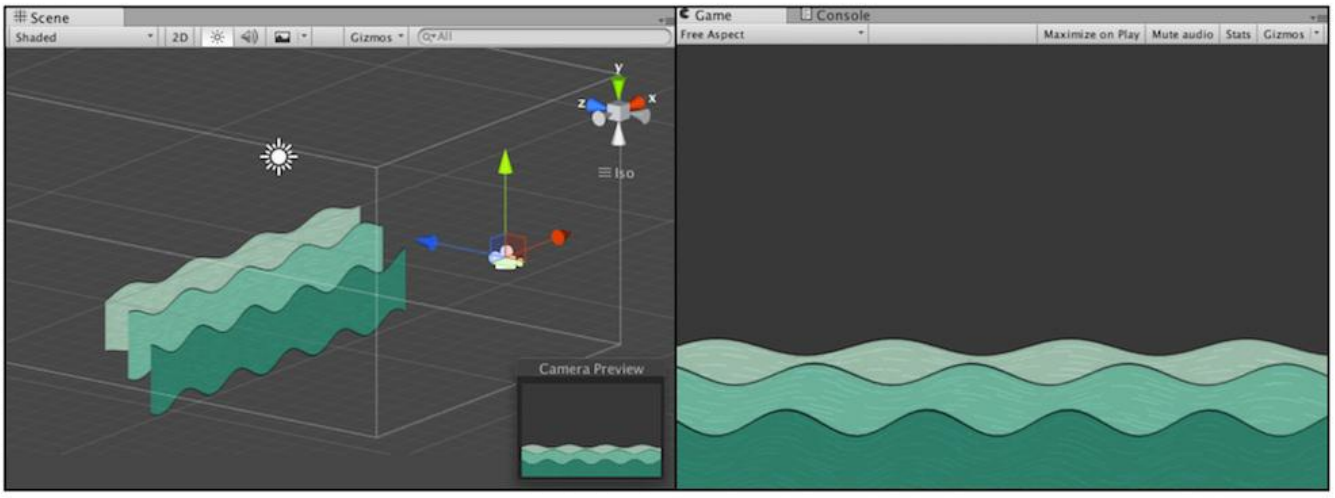
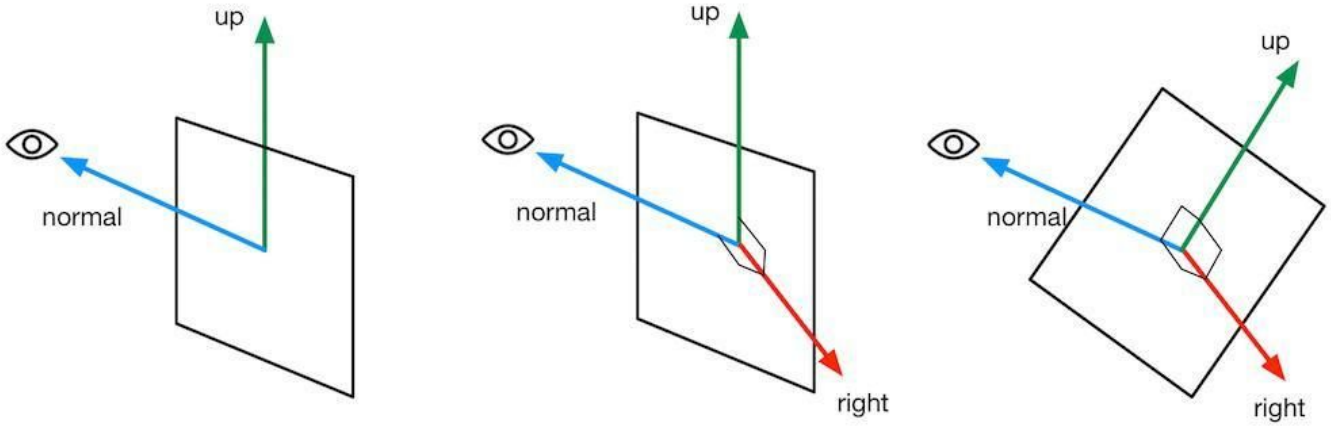


Figure 11.4 using vertex animation to simulateriver2D



the normalFigure11.5fixed (always pointing direction of view) , the process of calculating three orthogonal yl billboard art

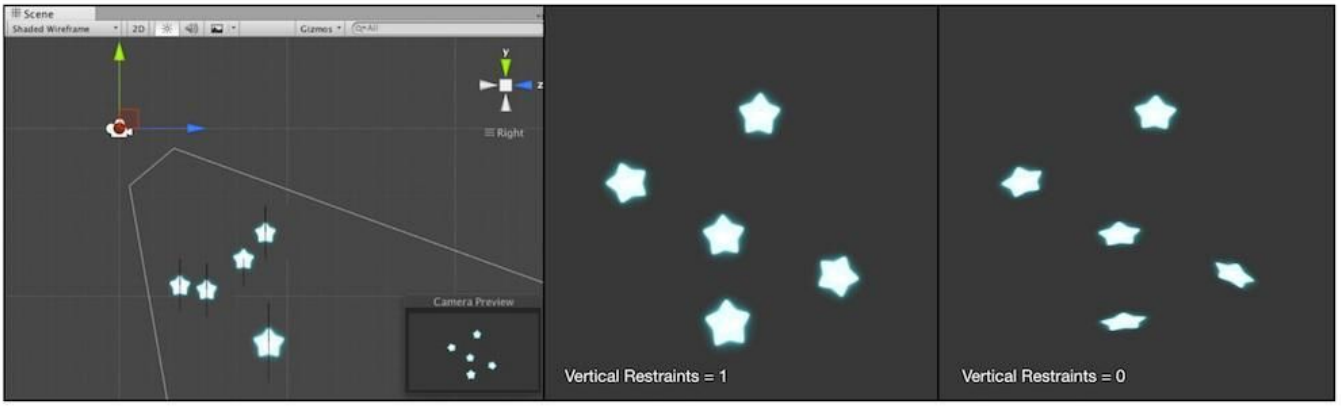
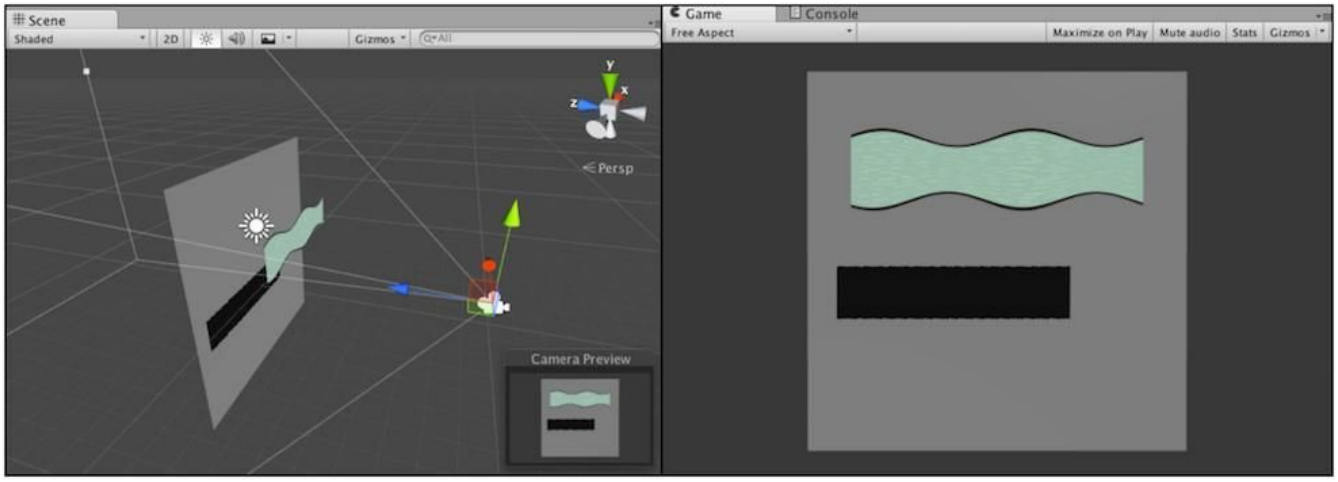


FIG11.6 billboard effect. The left panel shows the positional relationship between the camera and five billboards, cameras are observed them from the oblique direction. The middle shows the effect when the property is Vertical Restraints 1, i.e., fixed viewing angle of the normal

direction is obtained, it can be seen, all the billboards are fully facing the camera. The right panel shows the results when the Vertical Restraints property is 0, i.e., the direction of the fixed point is (0, 1, 0) is obtained, it can be seen, although the maximum billboard

facing the camera, but on the pointthe direction has not changed



when11.7 when vertex animation, if you still use the built-ShadowCaster Pass to render shadows, you may get the wrong shadows

11.8 using a custom ShadowCaster Pass deformation of the object to draw the correctshadow

screenChapter 12effect after treatment

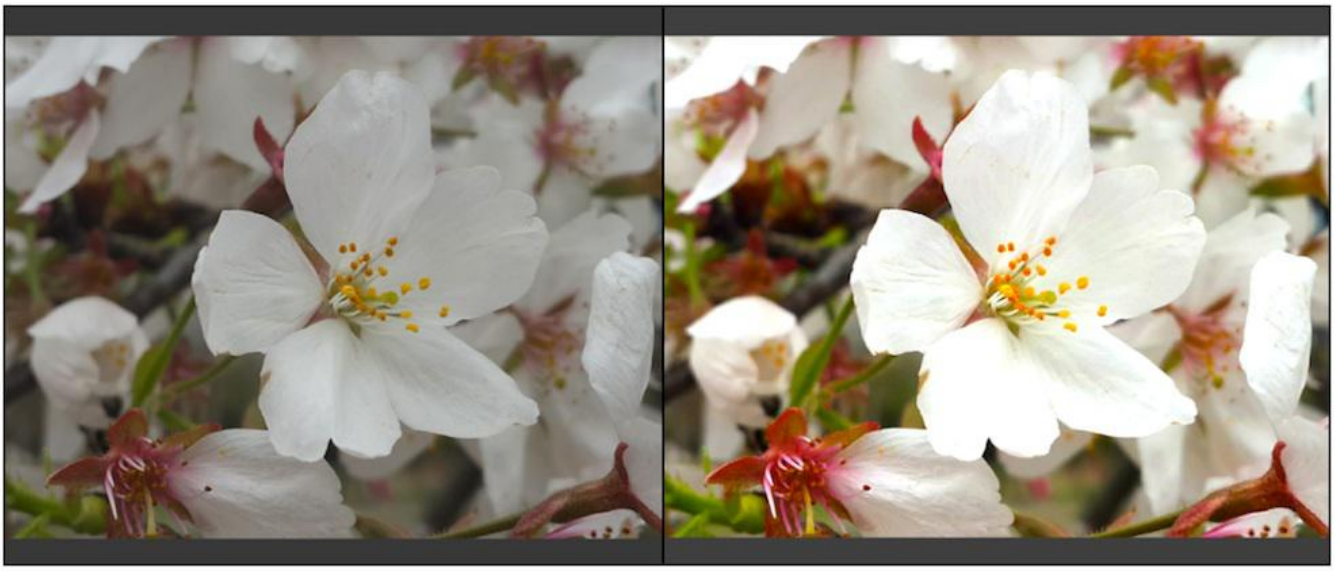
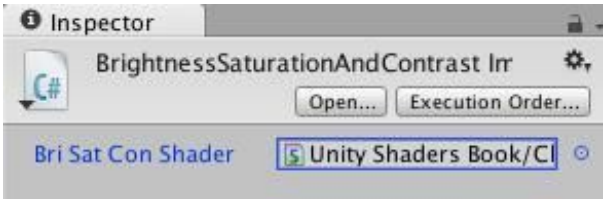
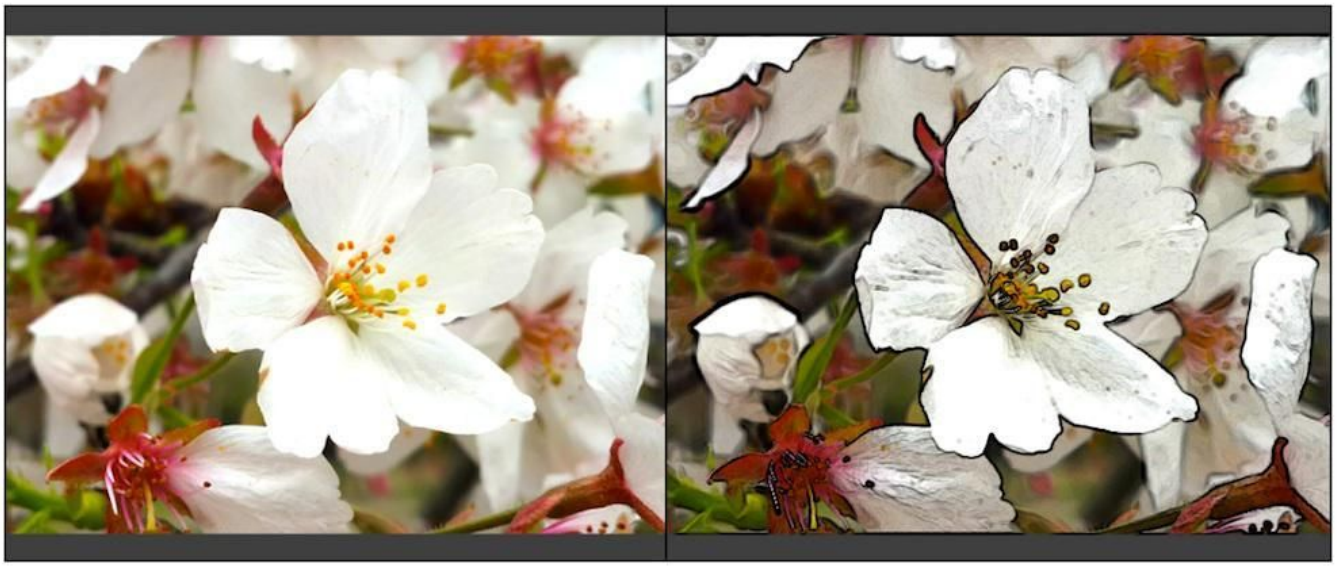


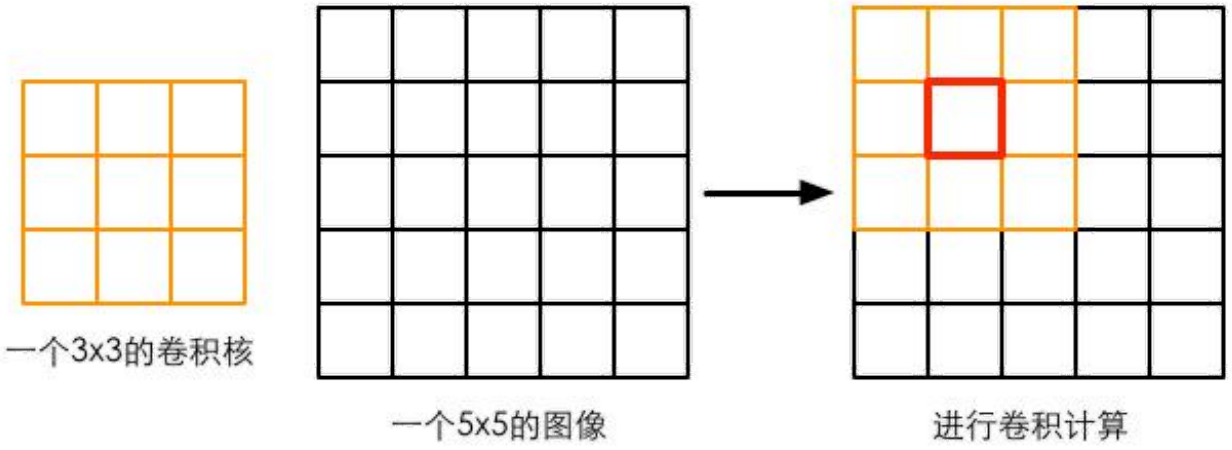
FIG12.1 left: original effect. Right: adjusting the brightness (value 1.2)afterthe saturation(value 1.6) and contrast (value 1.2)



12.2script set to a default value Shaderrenderings:



12.3 left in FIG12.2 The results obtained. Right: the effect of edge detection



convolution with a convolution kernel 12.4. Using a 3 × 3 convolution size matching a size of 5 × 5 performs image convolution, when the calculation map corresponding to a block convolution result of red pixels, we first placed in the center of the convolution kernel of pixel position, then flip the nuclear core in turn calculates a product of each element and the image pixel values which covers and summed to obtain the newpixel value

Roberts

-1	0
0	1

Gx

0	-1
1	0

Gy

Prewitt

-1	-1	-1
0	0	0
1	1	1

Gx

-1	0	1
-1	0	1
-1	0	1

Gy

Sobel

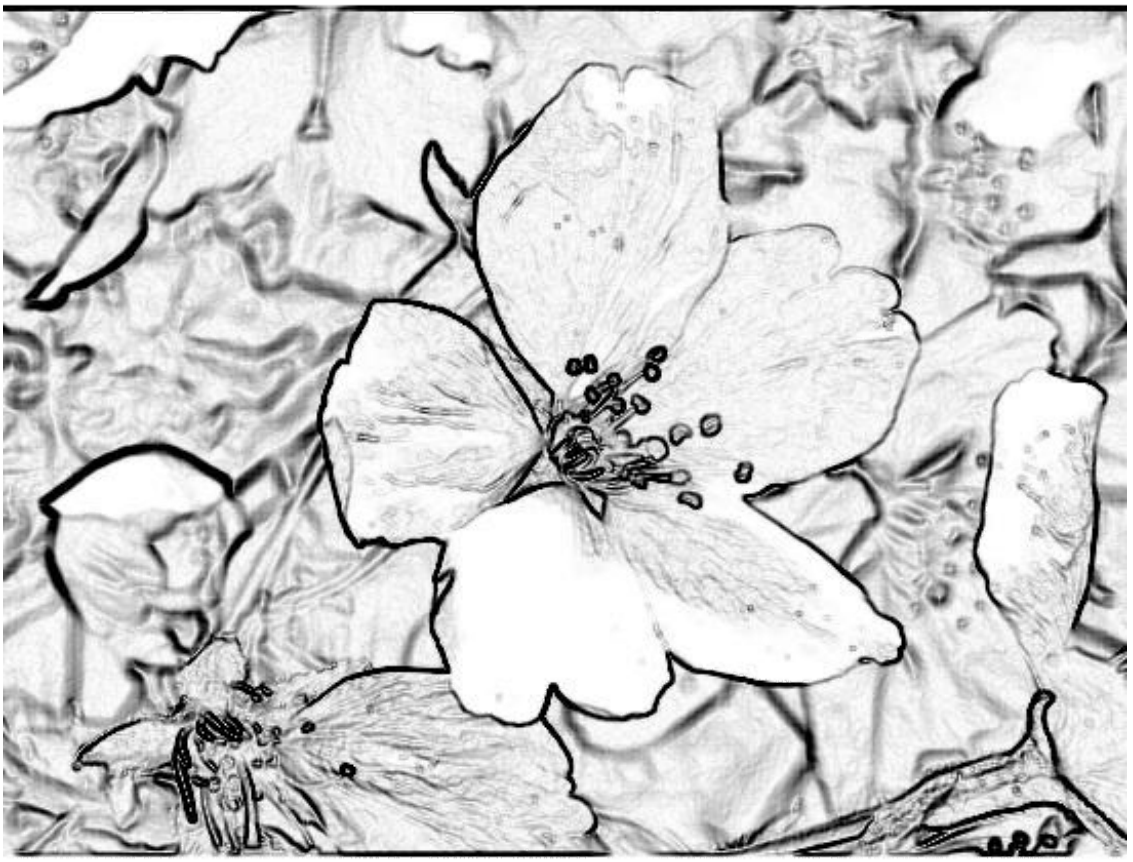
-1	-2	-1
0	0	0
1	2	1

Gx

-1	0	1
-2	0	2
-1	0	1

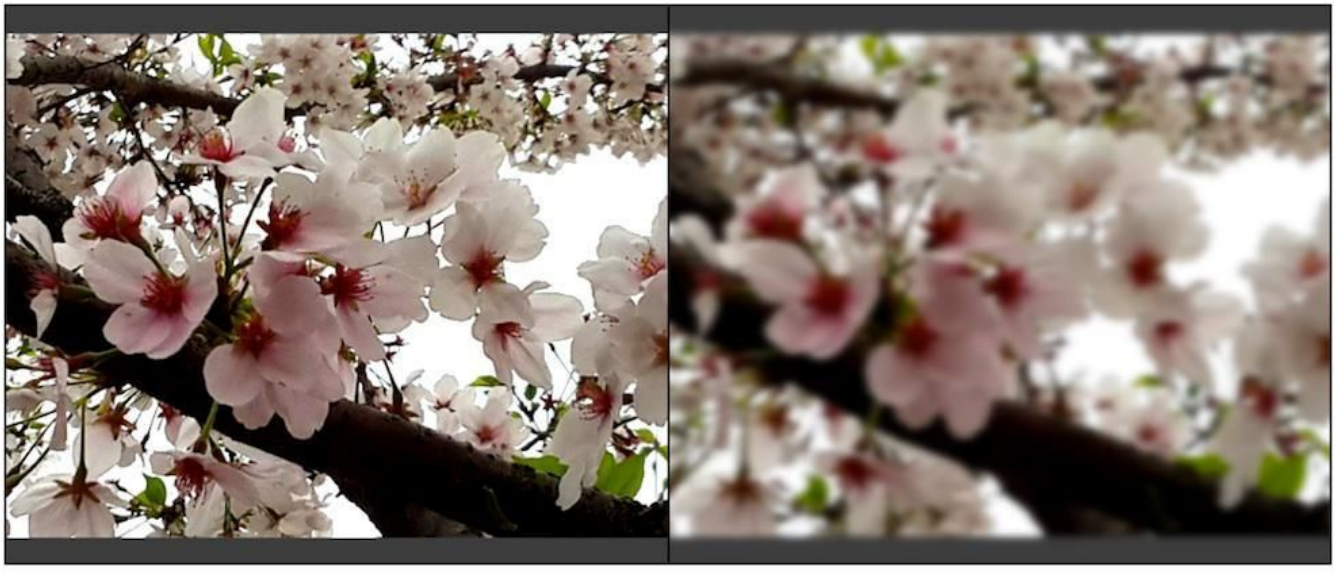
Gy

12.5 FIG three common edge detector

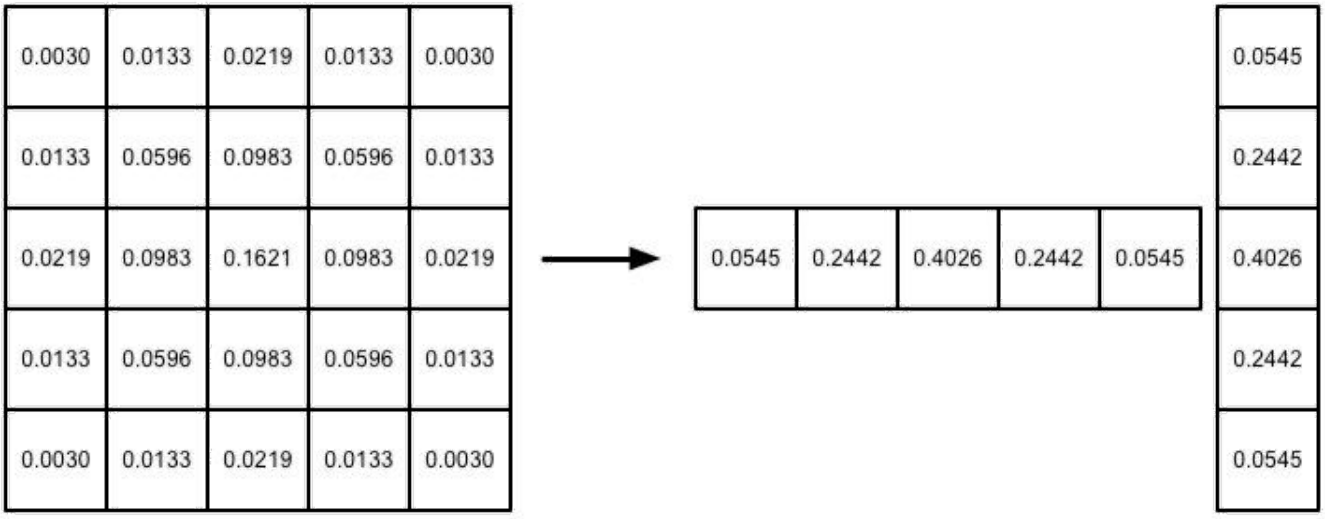


screen

show in FIG edge 12.6

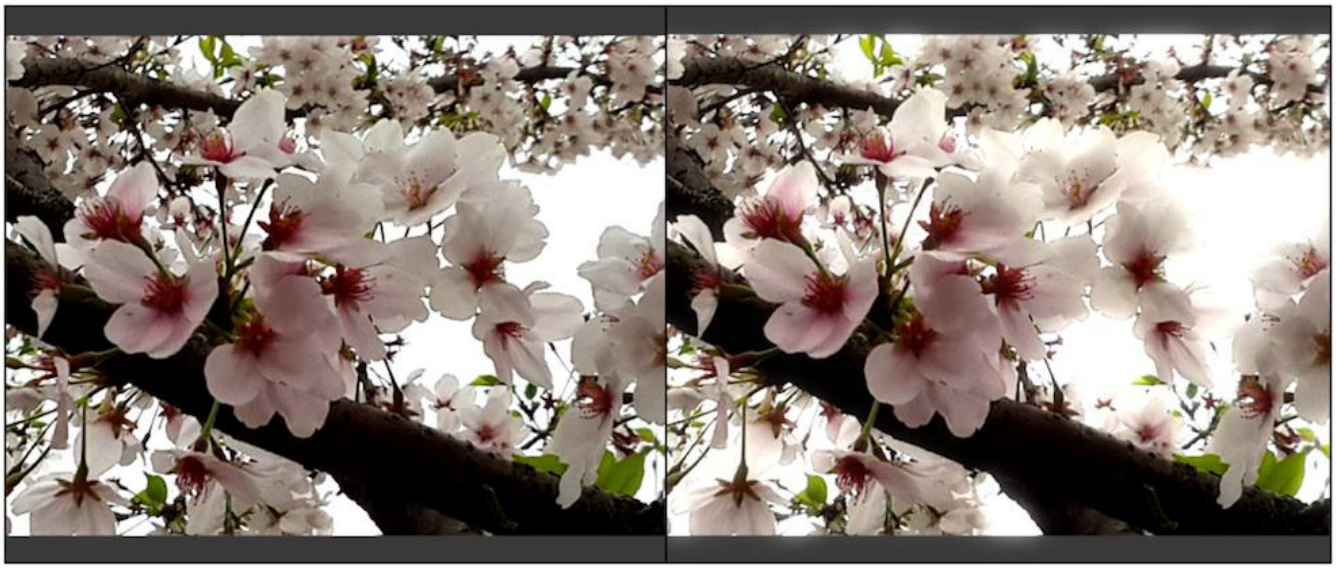


rendering12.7 left: original effect. Right:Gaussian blur after



FIGa 5 × 5 12.8 Gaussian kernel size. The left panel shows the standard deviation of the Gaussian kernel 1 right weight distribution.we can put this two-dimensional Gaussian kernel split into two one-dimensional Gaussian kernel (right)

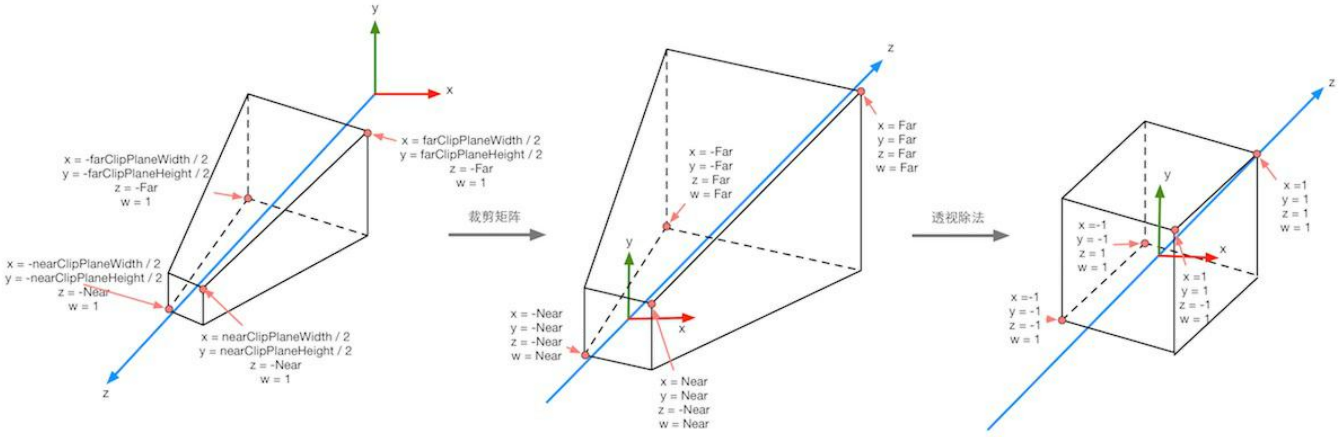
Bloom effectFigure 12.9 animated short film "Elephants Dream". Diffuse light through the door into the dark area surroundingin



the leftFIG 12.10: original effect. Right: the effect of treatment Bloom

12.11 left: original effect. Right: applying motion blur effect after

Chapter 13 normals and textureusing a depth



mapsin the perspective projection 13.1, the first projection matrix is scaled vertices. After homogeneous division, perspective projection conversion will be cut to a cube of space. FIG 4 are denoted by the key through the results ofprojectiontransformation

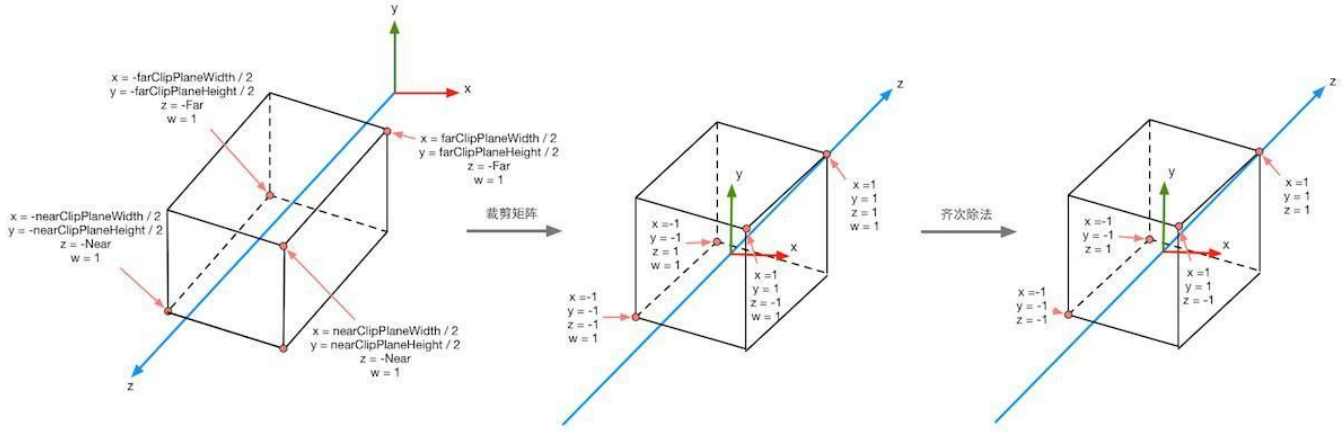


FIG13.2 in orthogonal projectionmatrix,the projection matrix is scaled vertices. After homogeneous division, orthogonal projection of the clip space will be transformed into a cube. FIG noted in four key points after the result of the projection matrix transforms

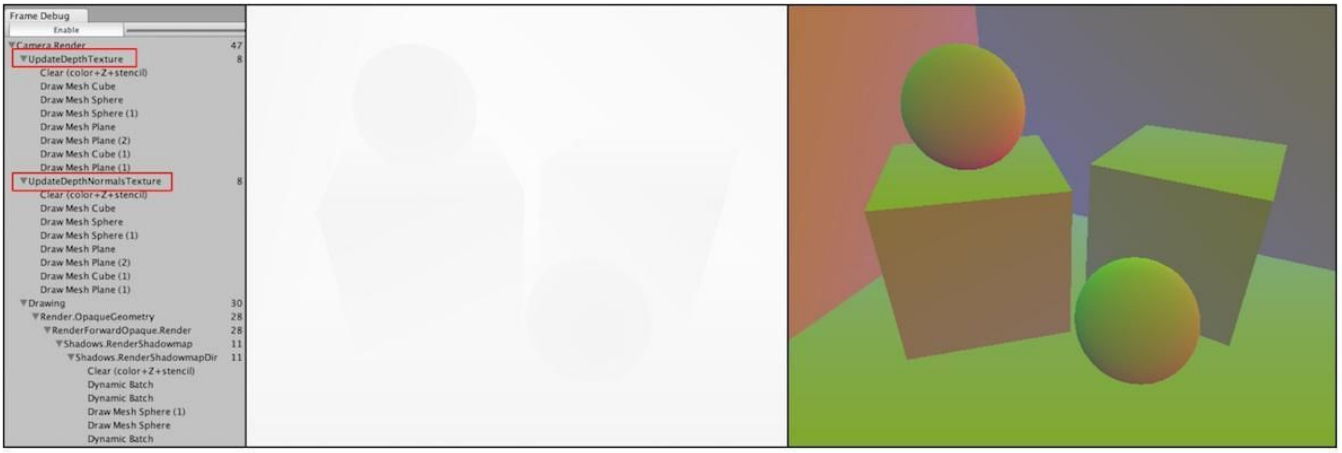
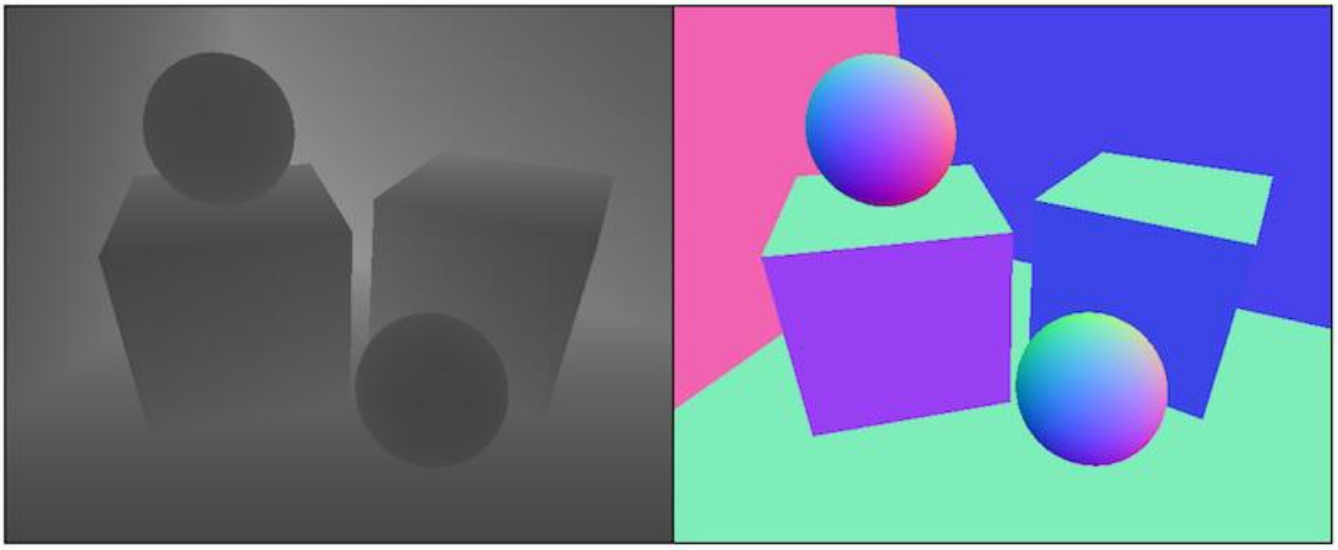


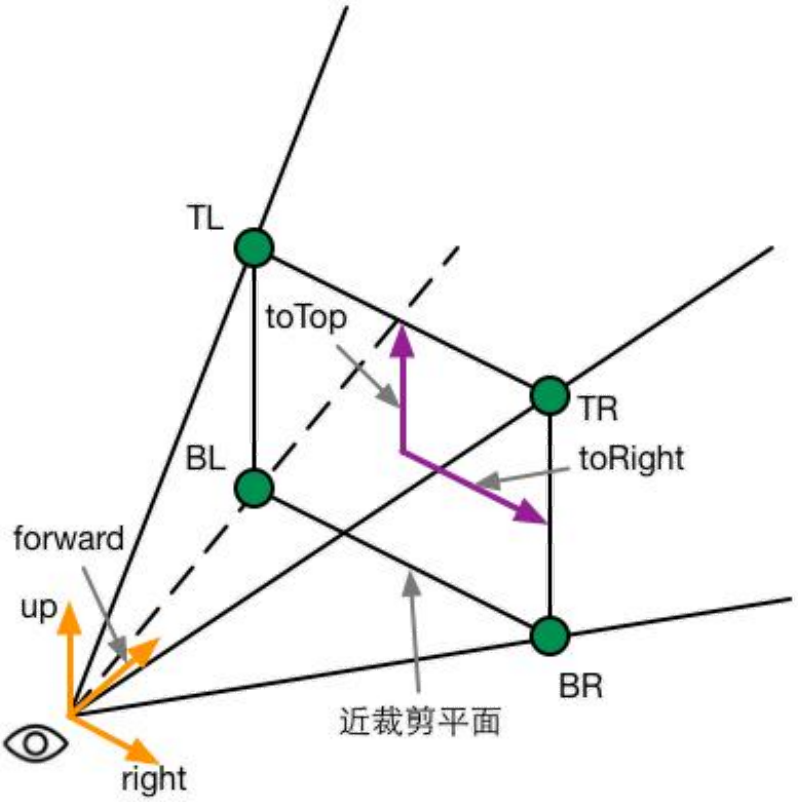
FIG 13.3 Frame Debugger depth texture view (left) and normal texture depth + (right). If you need to generate the current camera depth and normals texture, rendering the corresponding event in the frame of the debugger panel appears. The normal depth and textureby simply clicking the corresponding event you can view the resulting



maps13.4 left: depth texture in linear space. Right: decoded and mapped to [0, 1]

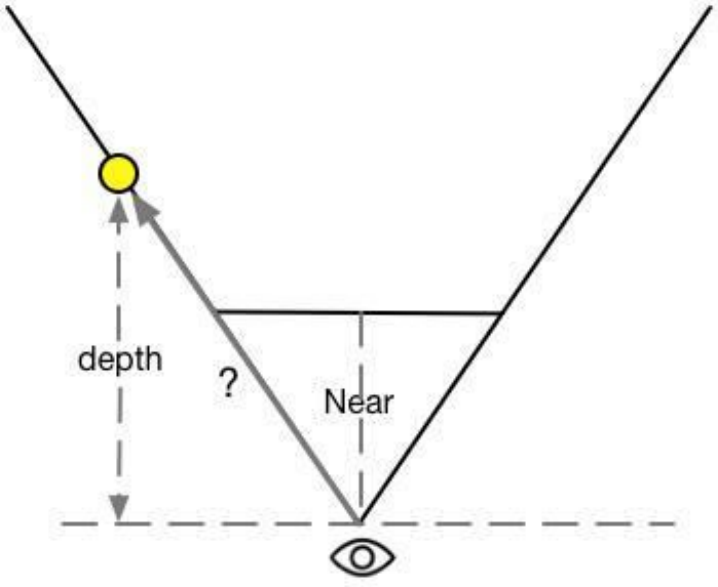
Perspective normal texturespacein the range of

mapleft13.5:original effect. Right: after adding the global effect of fog

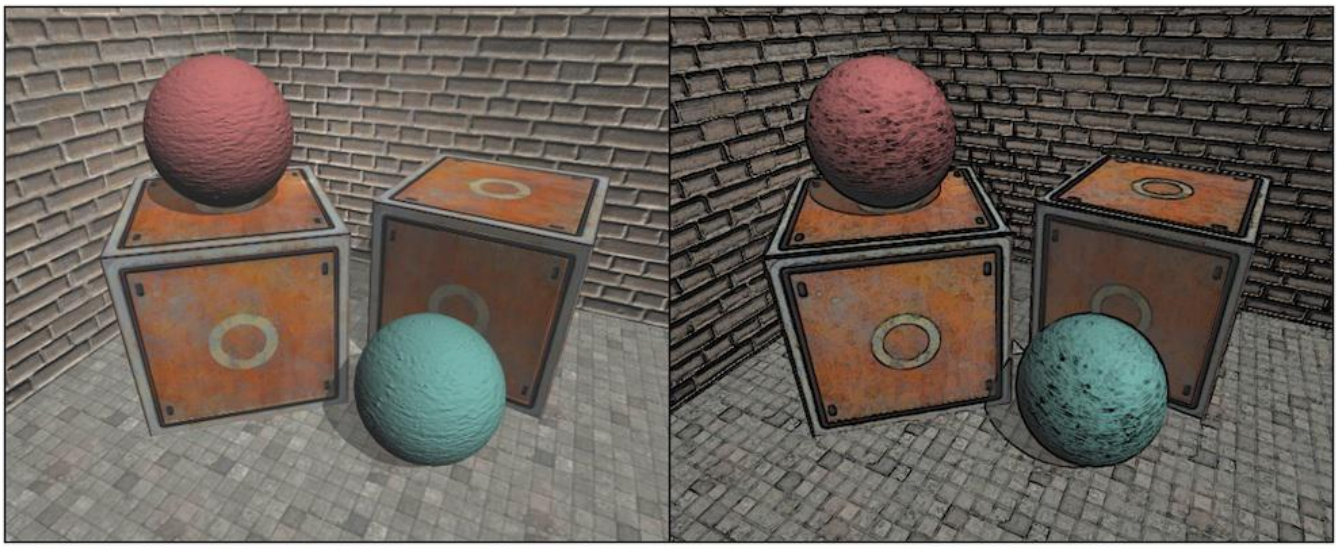


FIG

calculateinterpolatedRay13.6



depth valuesFIG13.7 sampled point not Euclidean distance to the camera



in FIG 13.8 Left: original effect. Right:a direct result of the color image edge detection

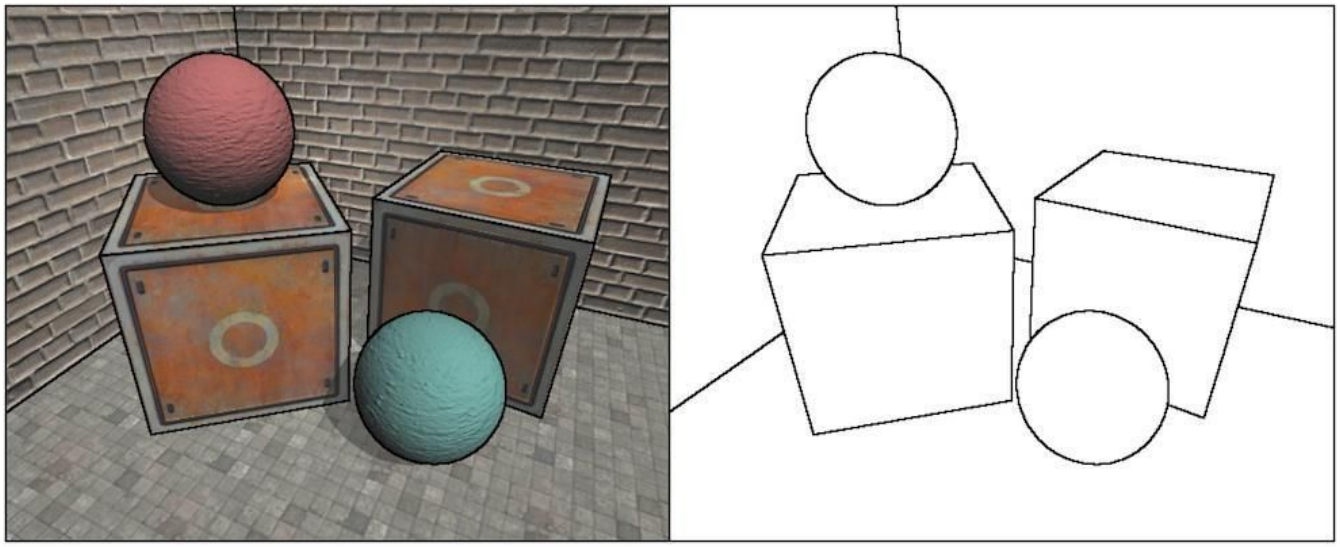
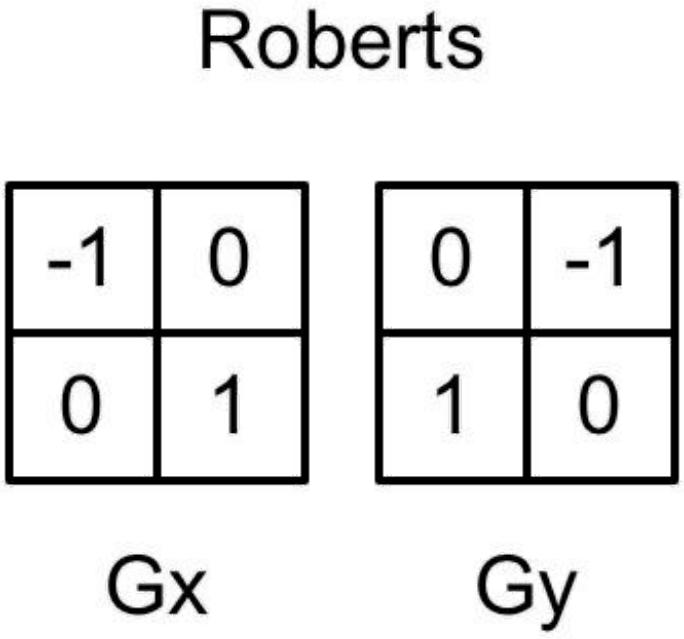


diagram ofmore robust 13.9 edge detection in the depth and normal texture. Left: the effect on the original stroke. Right: Show only stroke



renderings13.10

Roberts operator

Chapter 14Non-photorealistic

14.1renderinggame "Big God" (English name: Okami) game screenshots

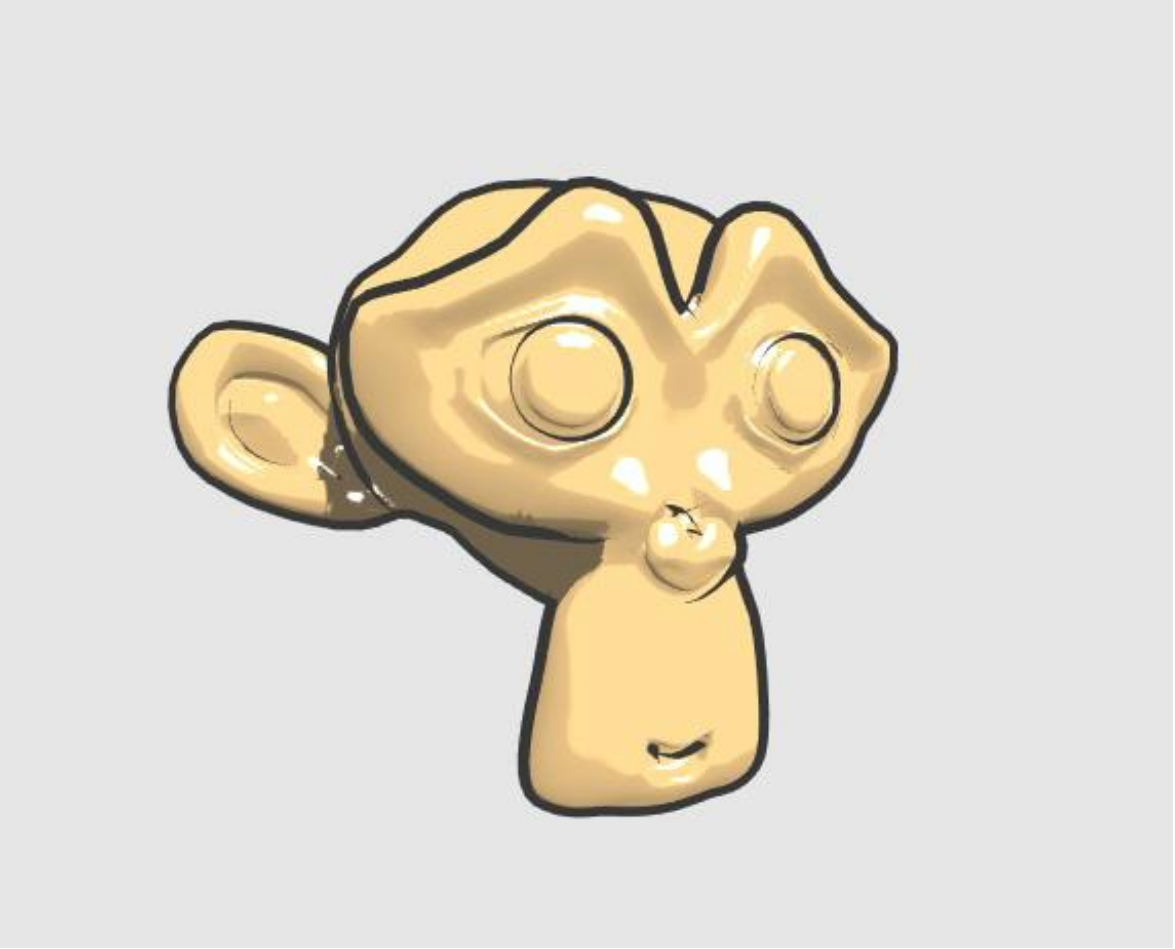
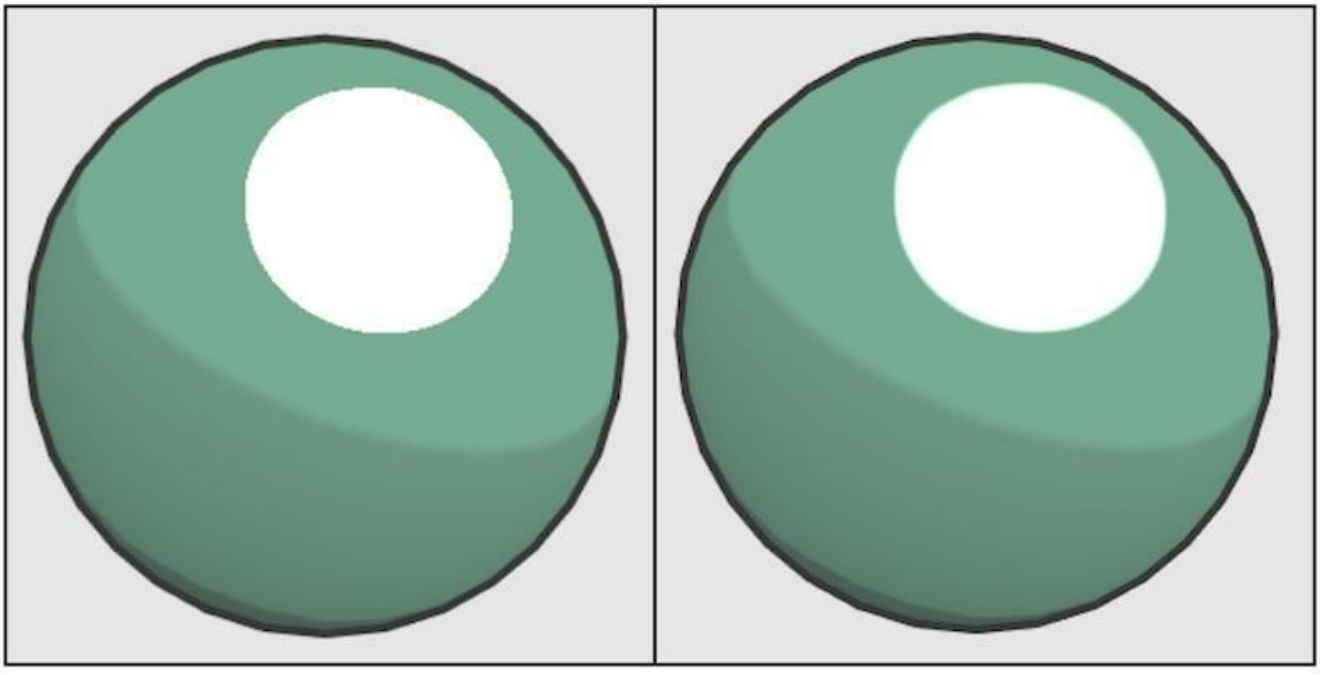
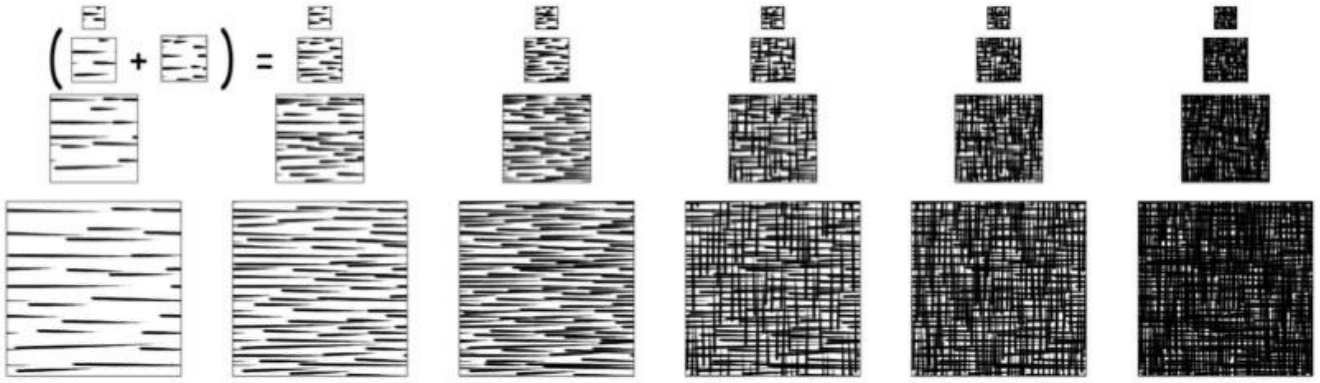


Figure 14.2 cartoon



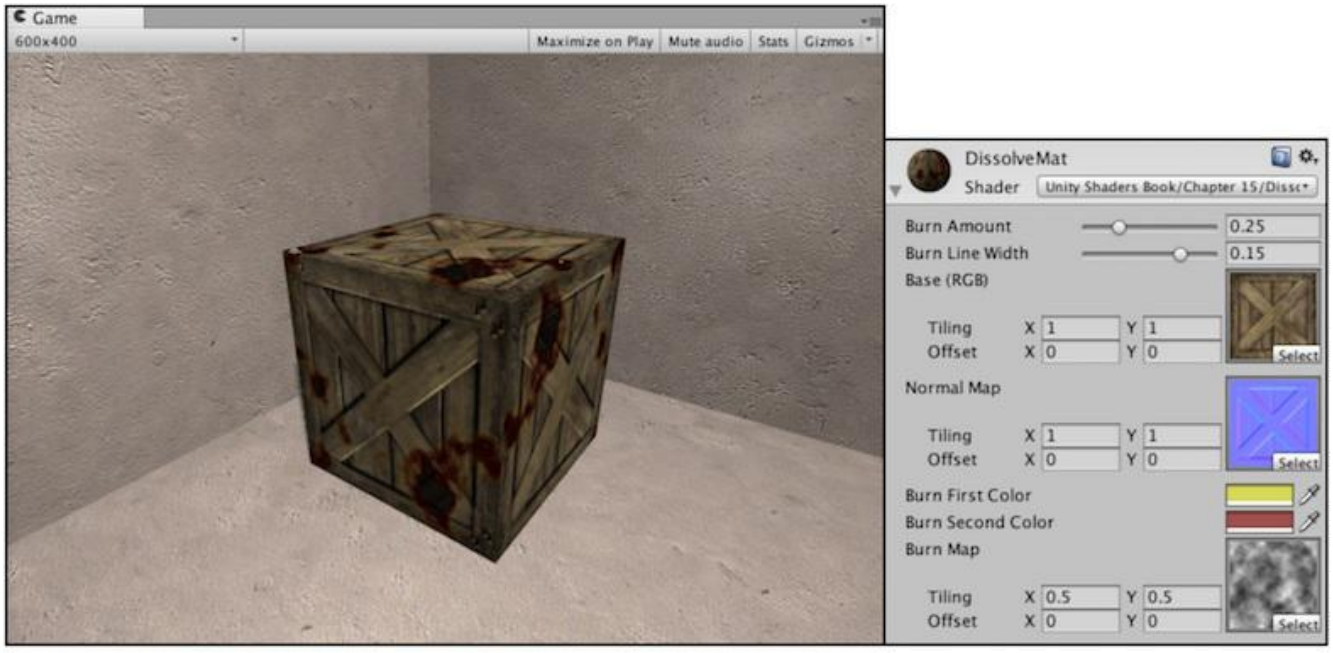
renderings14.3 Left: Not to highlight region antialiasing. Right: fwidth function using fora high light areaantialiasing



example14.4a TAM (source:. Praun E, et al Real

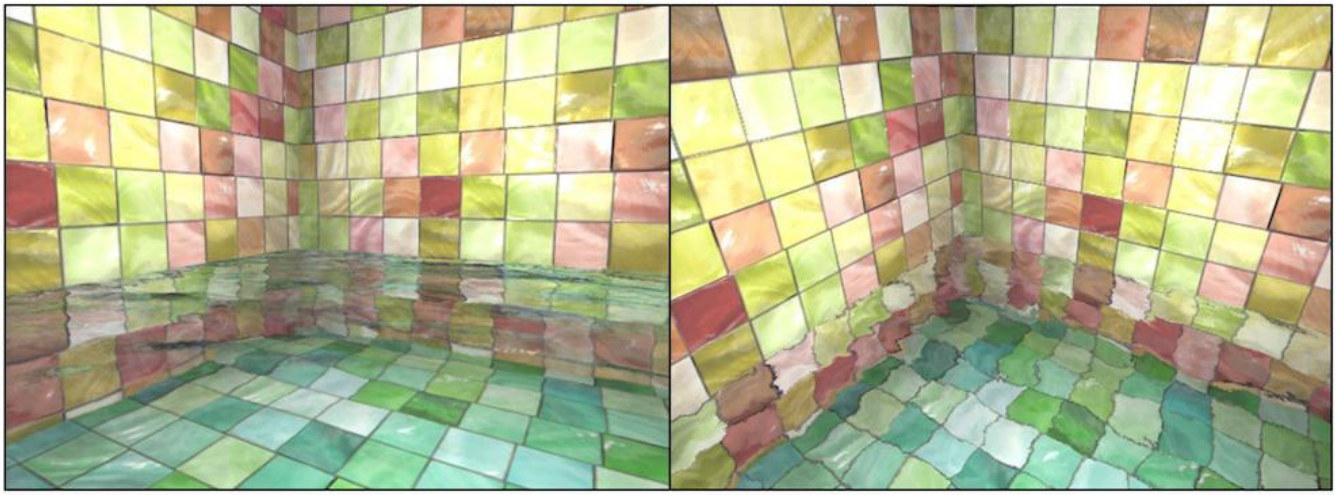
14.5 sketch style rendering

Chapter15 using the noise



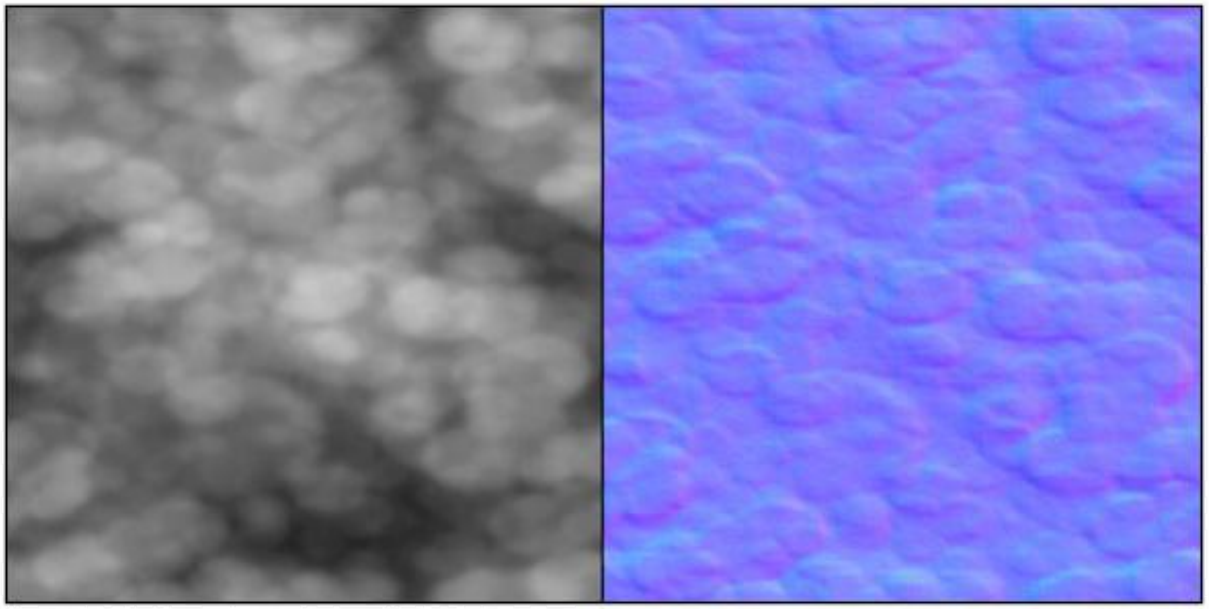
15.1 in FIG.box ablationrenderings

ablation 15.2 noise effect of using a texture

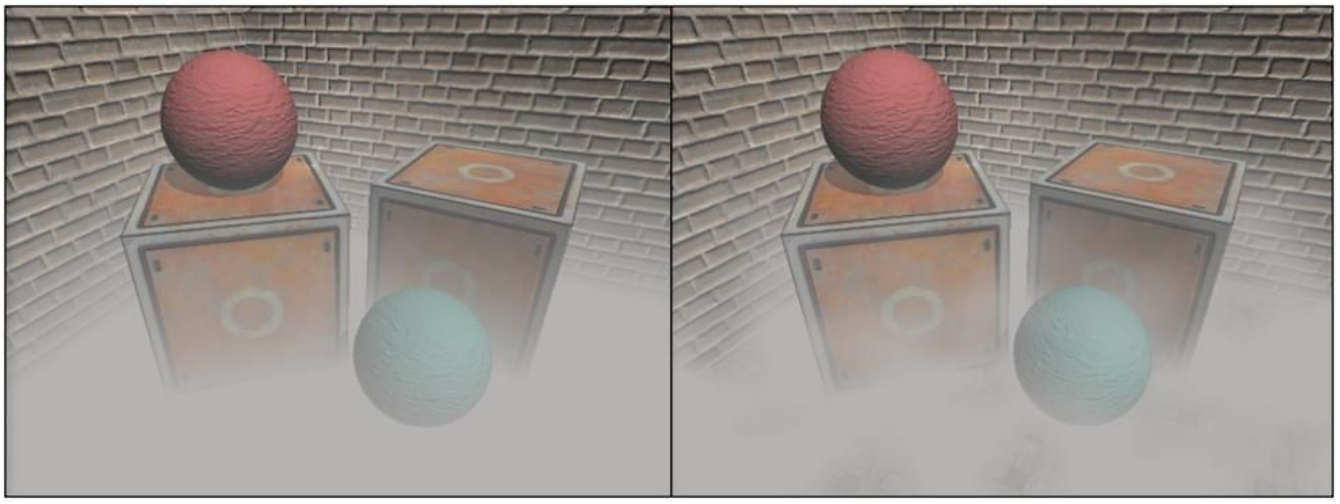


map comprising water 15.3 fluctuation effect of Fresnel reflection. On the left, the greater the angle between the viewing direction and the normal to the surface, the stronger the reflection effect. On the right, the greater the angle between the viewing direction and the normal to the surface, the stronger the effect refraction

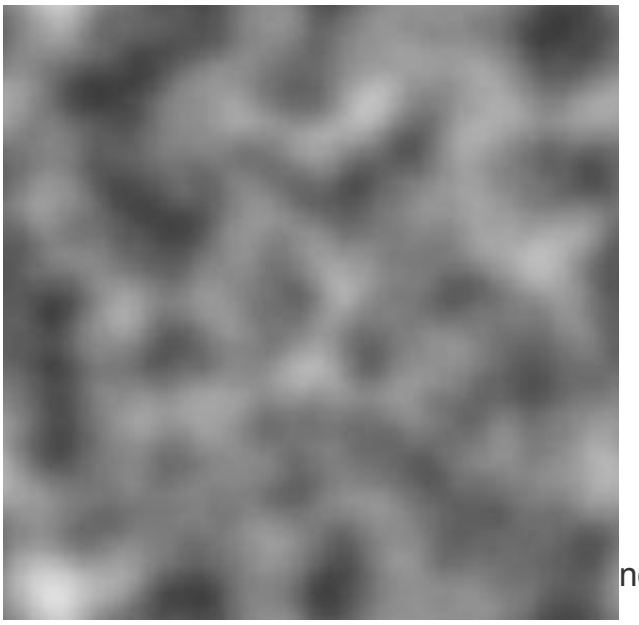
FIG15.4 Example cube textureused.



noise wave effect using texture 15.5maps Left: grayscale noise texture. Right: normal texture generated by the left

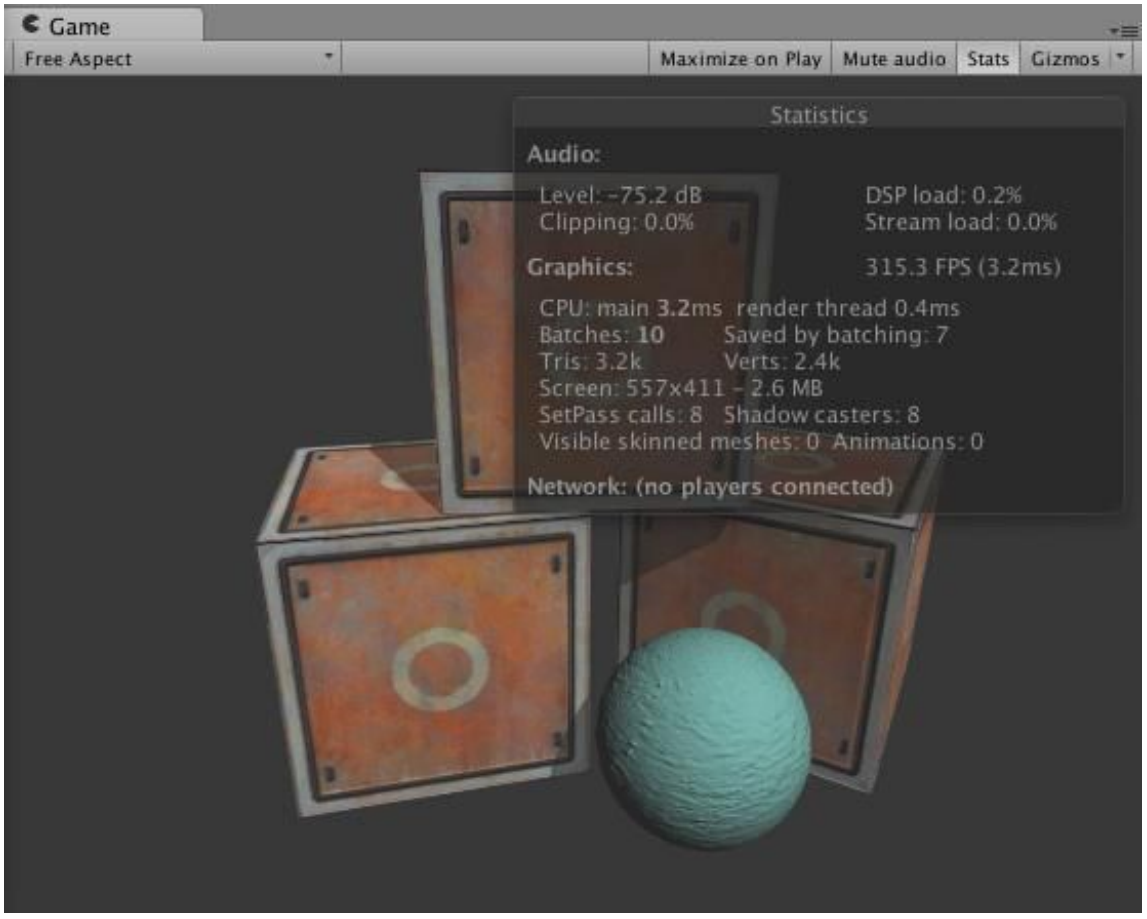


15.6 Left: uniform fog. Right:non-uniform fogafterusing the noise texture

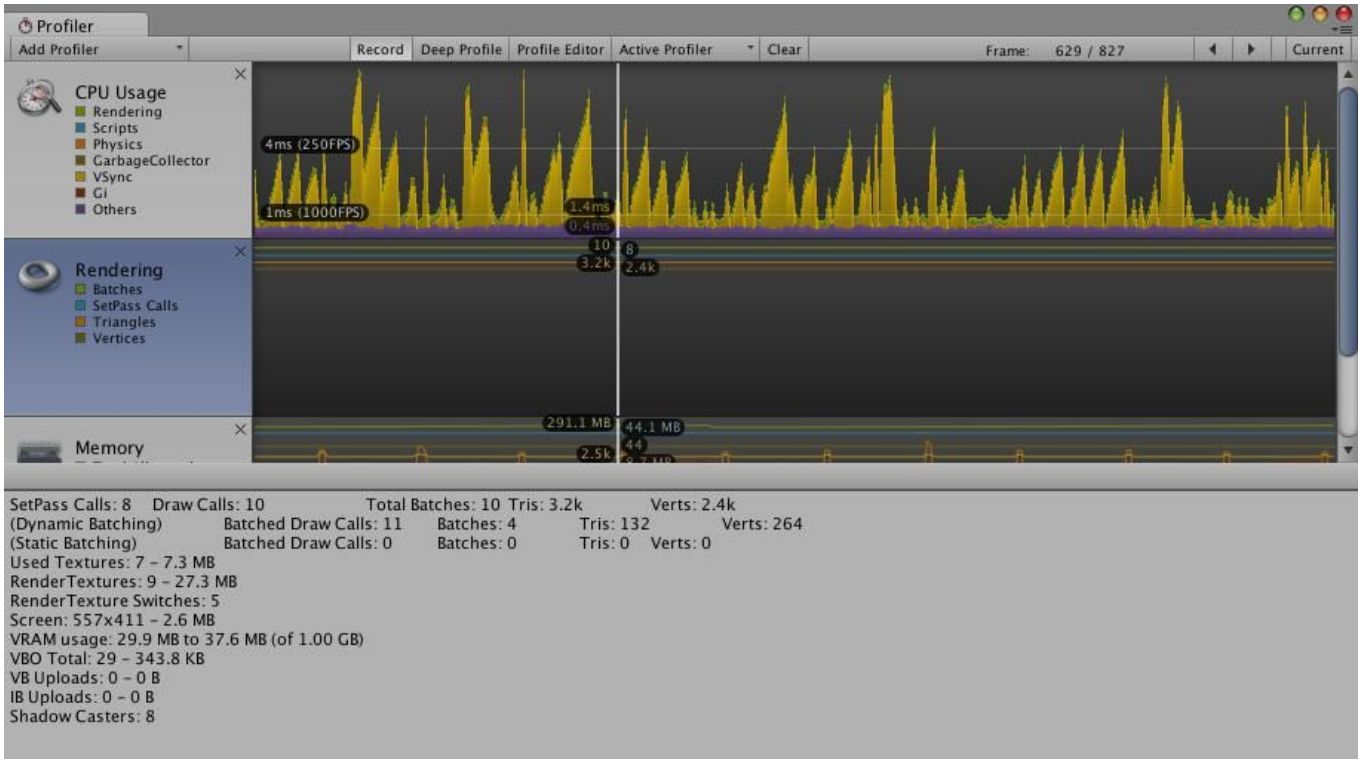


noise texture15.7sectionusedthe

renderingoptimizationChapterUnity16in

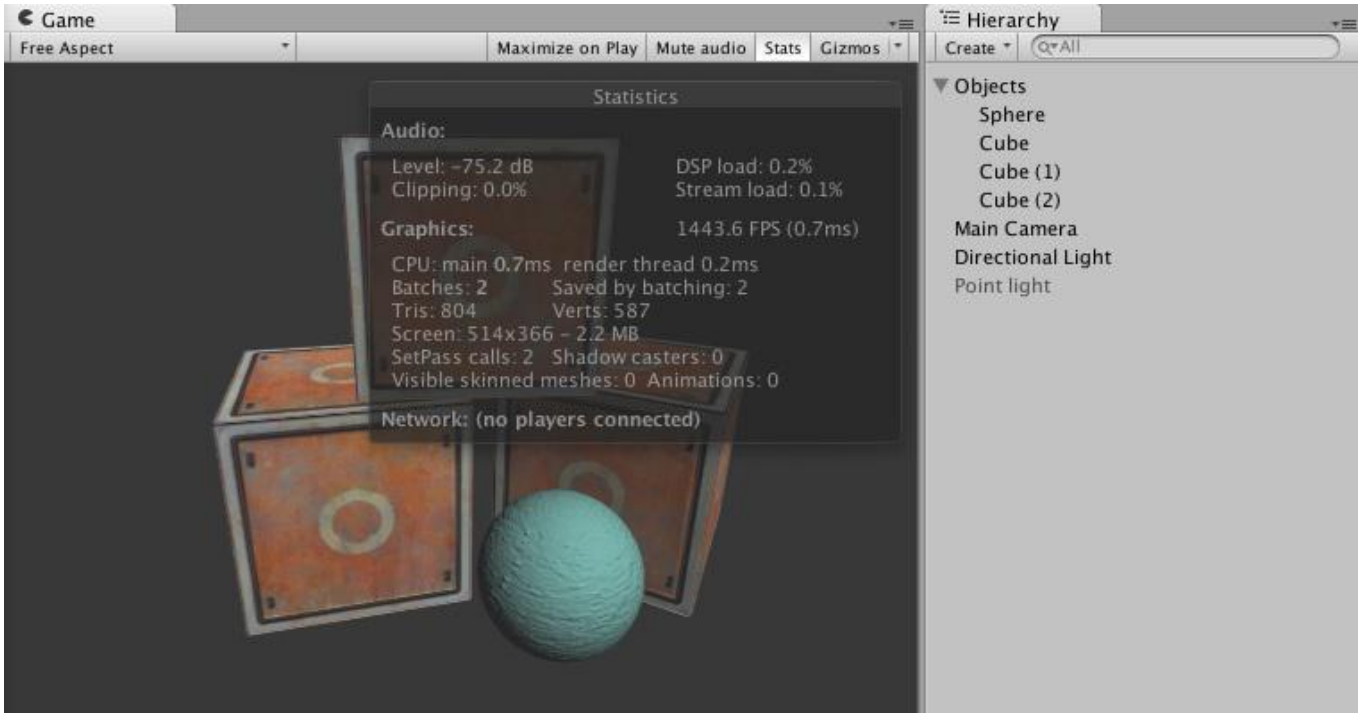


FIGstatistics window rendered16.1 Unity 5



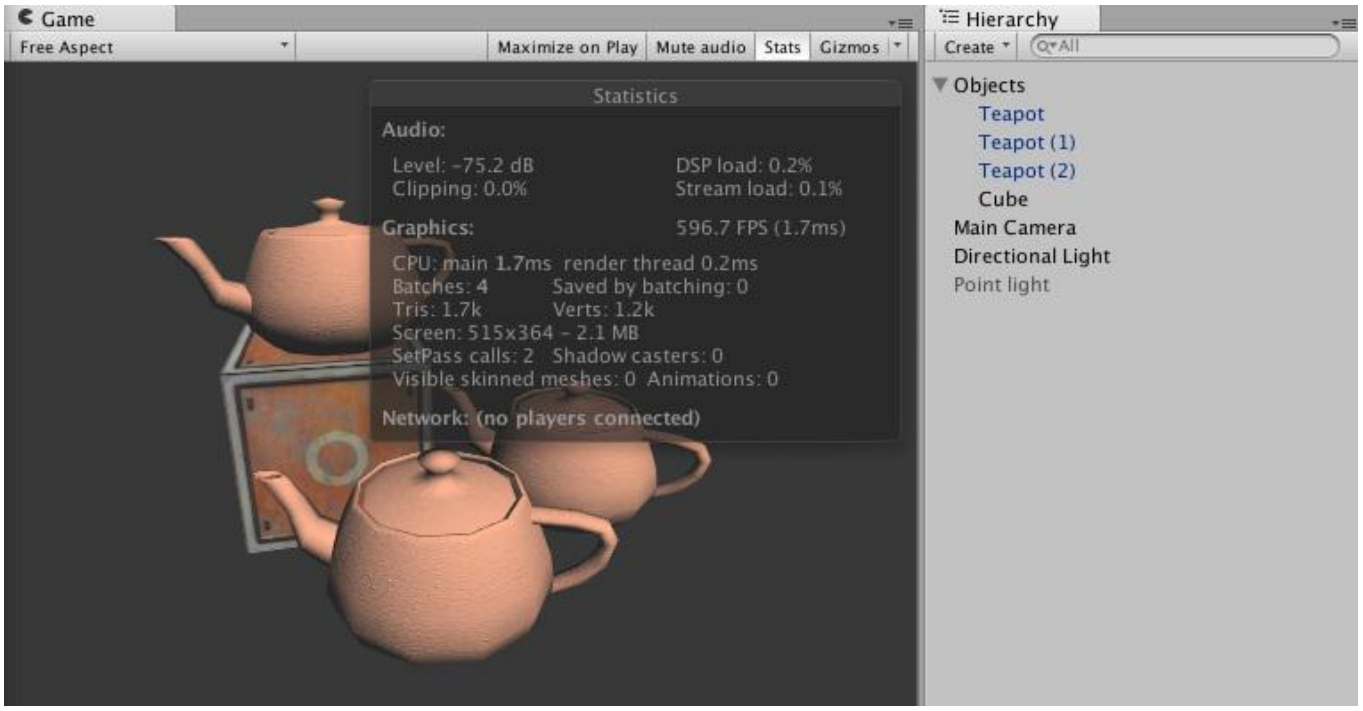
FIGrendering region Unity 16.2 Performance Analyzer to See more statistics on the rendering

frame in FIG 16.3debugger to view theindividualresults plotteddraw call



in FIG16.4dynamicbatch

over 16.5EffectsFIG dynamic light batchresults of



16.6static batch before rendering statisticsFIG.

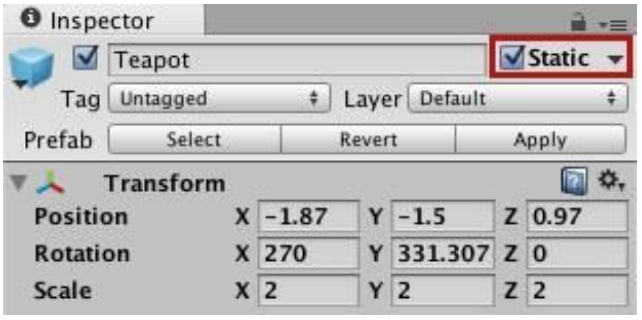


FIG16.7 static objects marked as

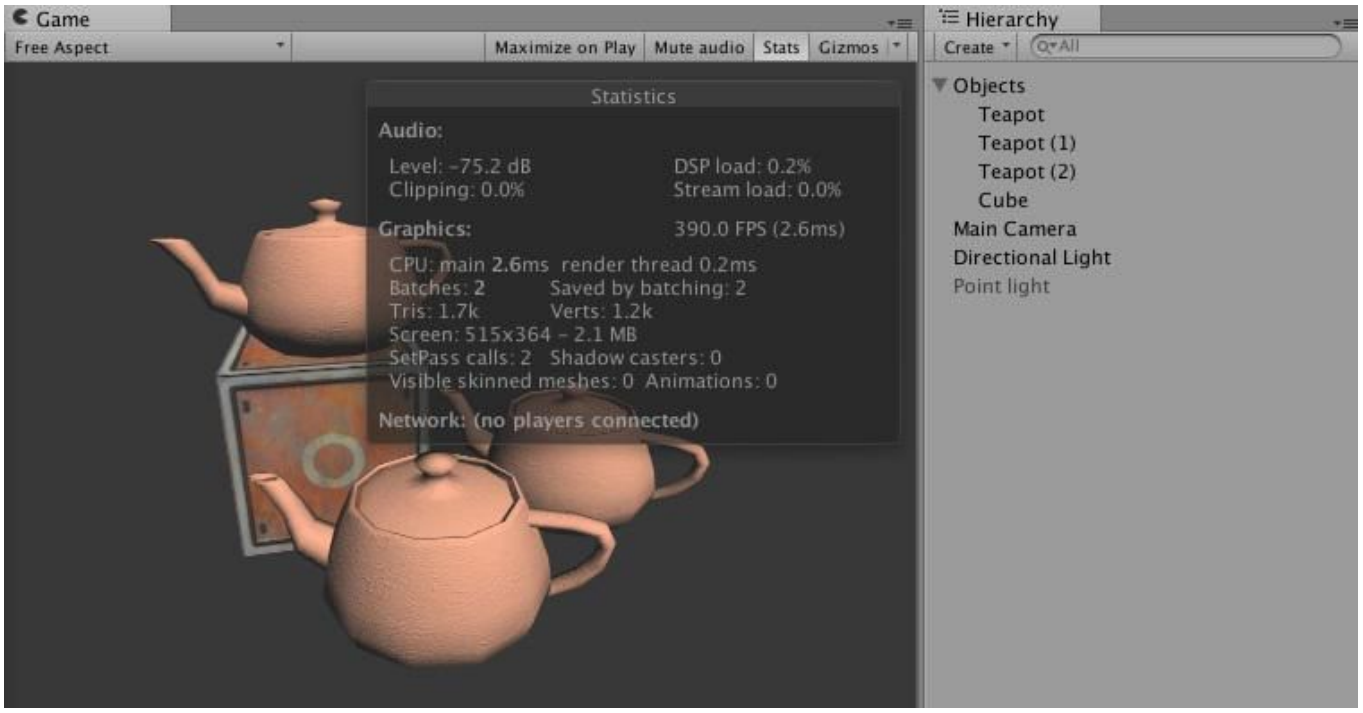


Figure 16.8static batch

Figure 16.9 staticbatch Unity will merge all are identified as "static" object

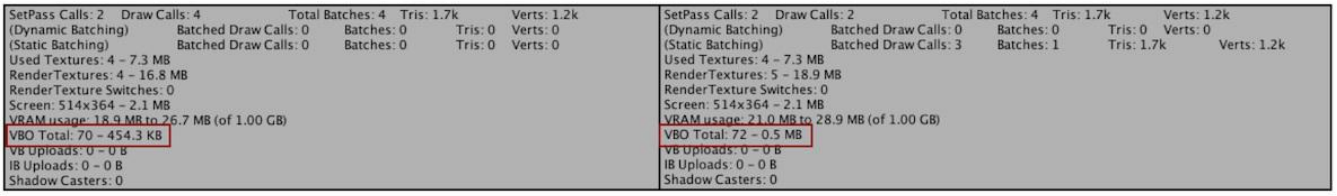
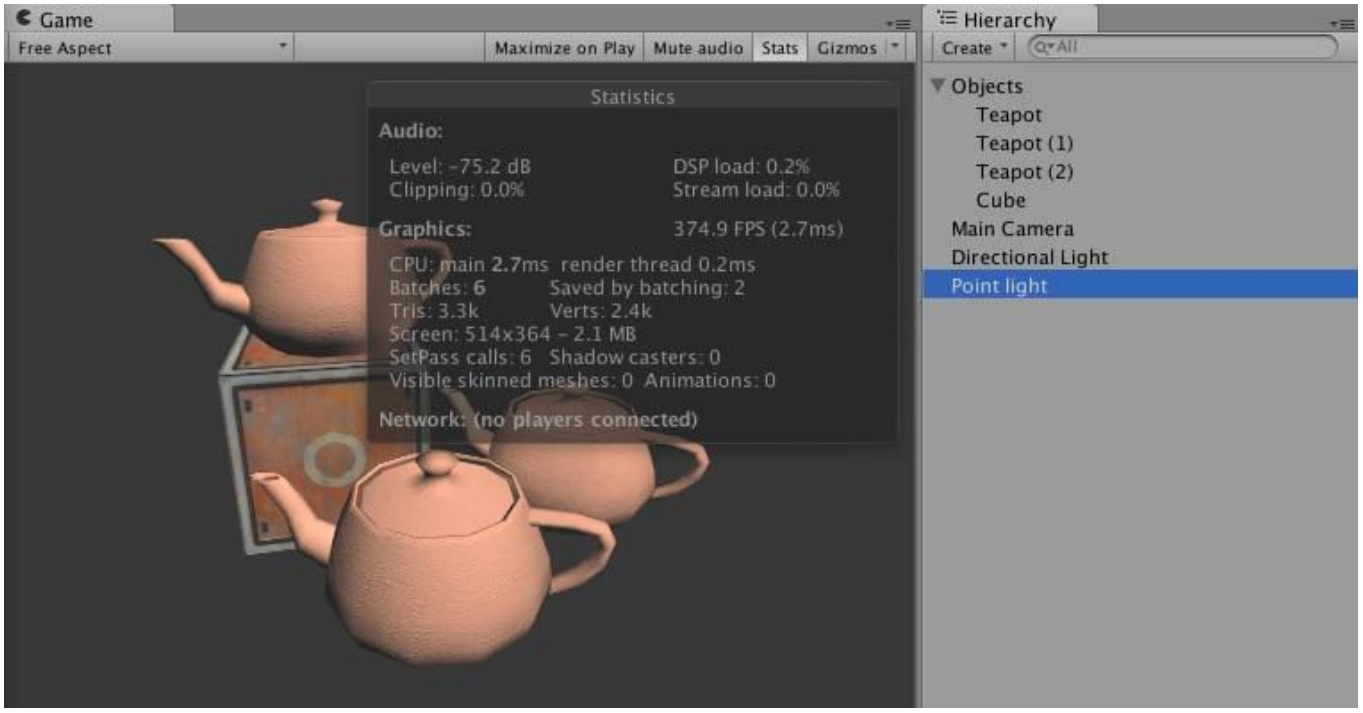


Figure 16.10 static batch will take up more memory. Left: statistical data before rendering static batch. Right: Static statistics rendered batch



viewof anotherprocessing 16.11 Pass light pixel by pixel is not static batch

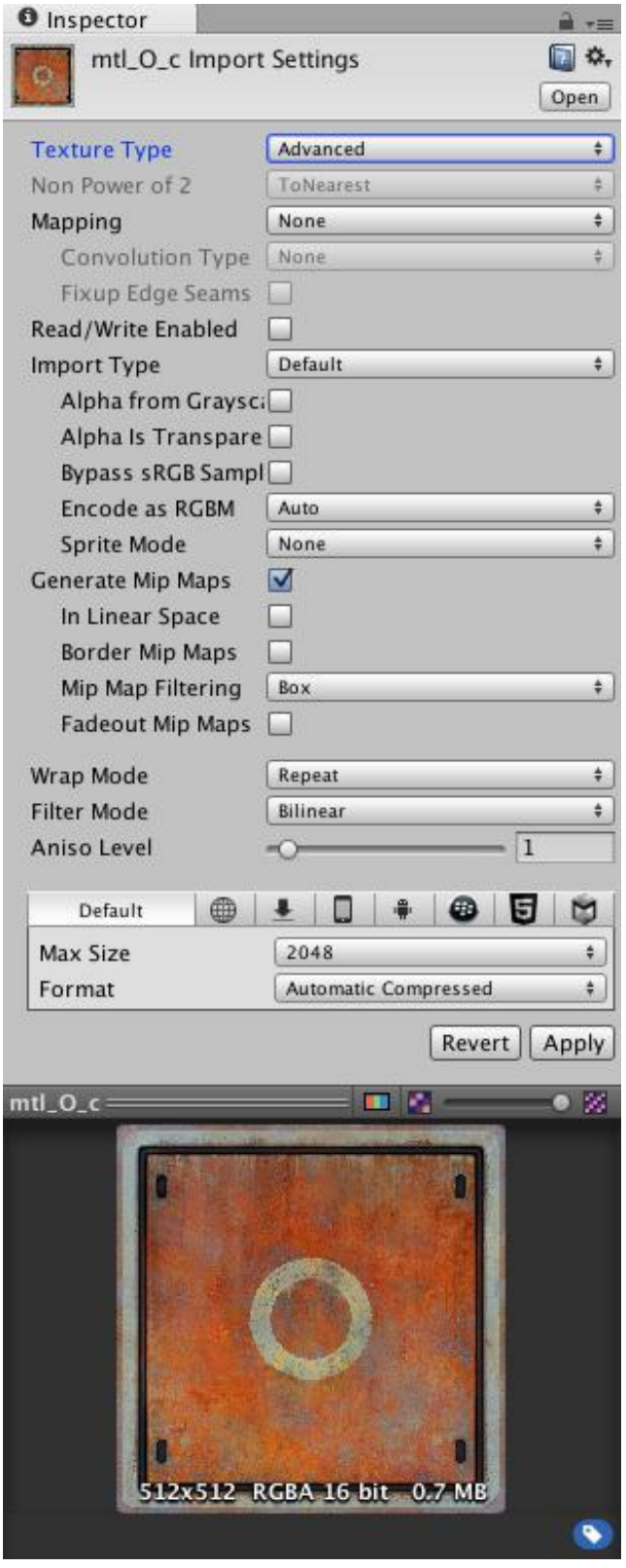
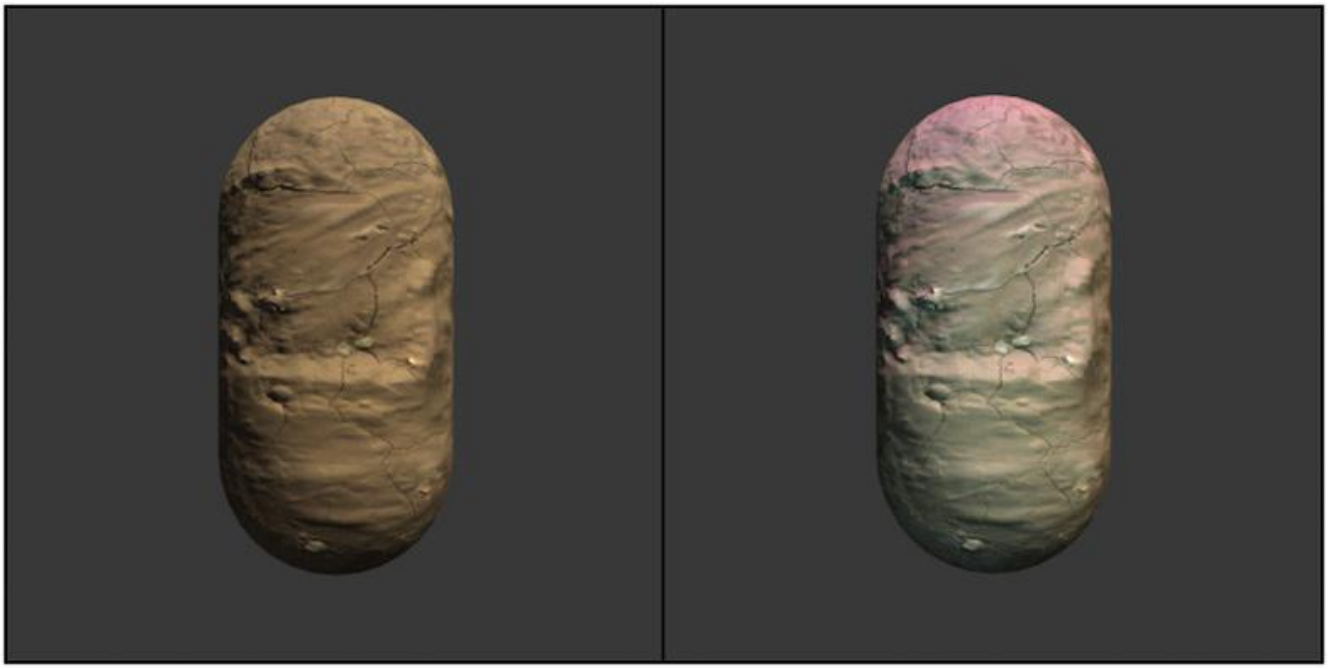

Inspector
mtl_O_c Import Settings
Open
Texture Type
Advanced
Non Power of 2
ToNearest
Mapping
None
Convolution Type
None
Fixup Edge Seams
Read/Write Enabled
Import Type
Default
Alpha from Graysc
Alpha Is Transpare
Bypass sRGB Sampl
Encode as RGBM
Auto
Sprite Mode
None
Generate Mip Maps
In Linear Space
Border Mip Maps
Mip Map Filtering
Box
Fadeout Mip Maps
Wrap Mode
Repeat
Filter Mode
Bilinear
Aniso Level
1
Default
Max Size
2048
Format
Automatic Compressed
Revert
Apply
mtl_O_c
512x512 RGBA 16 bit 0.7 MB

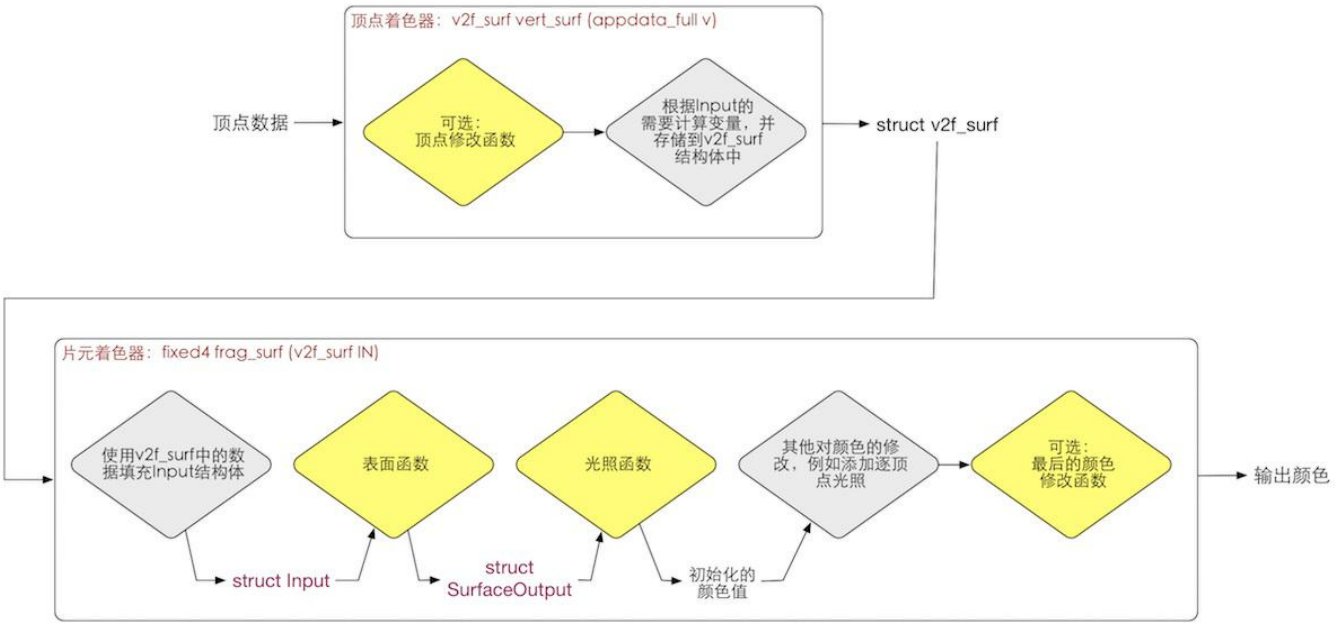
FIG advanced setting panel texture 16.12 Unity

Chapter17 Surface Shader Quest



17.1FIG surface shader examples. Left: effect under a parallel light. Right: after addition of a point light source (blue) and a spotlight (purple)renderings

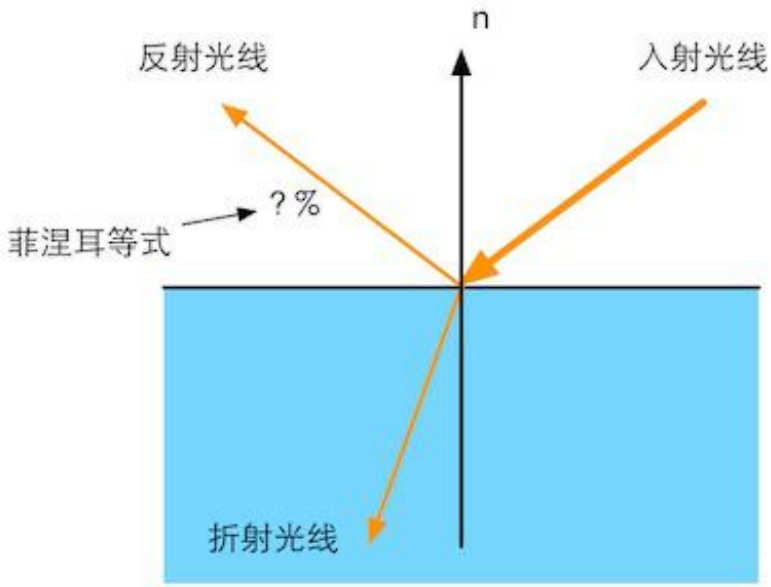
17.2 viewing surface shader code generated



rendering calculations 17.3 shader pipeline surface. Yellow: You can customize the function. Gray: calculating step Unityautomatically generated

Figure 17.4model was expanded along the vertex normals. Left: before expansion. Right:expansion

Chapter 18 based on the physicalofrendered



over 18.1 at the boundary, the refractive index of the light into two directions will mutation

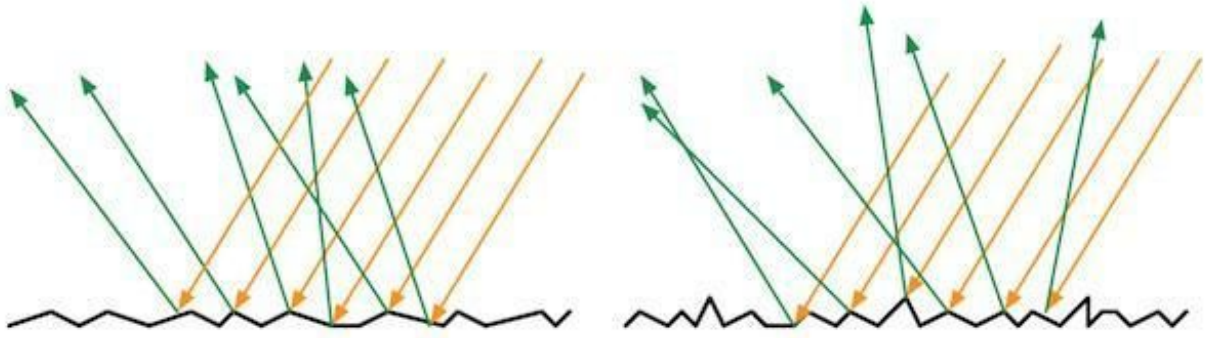


FIG 18.2 Left: normal small changes in the plane of a smooth surface of the micro, the reflected light the direction of change is smaller. Right: the micro-roughened surface normal of the plane of the larger changes, changes in the direction of the reflected light is greater

micro-surface light refracted 18.3 FIG. The refracted part of light is absorbed and scattered to the outside portion of

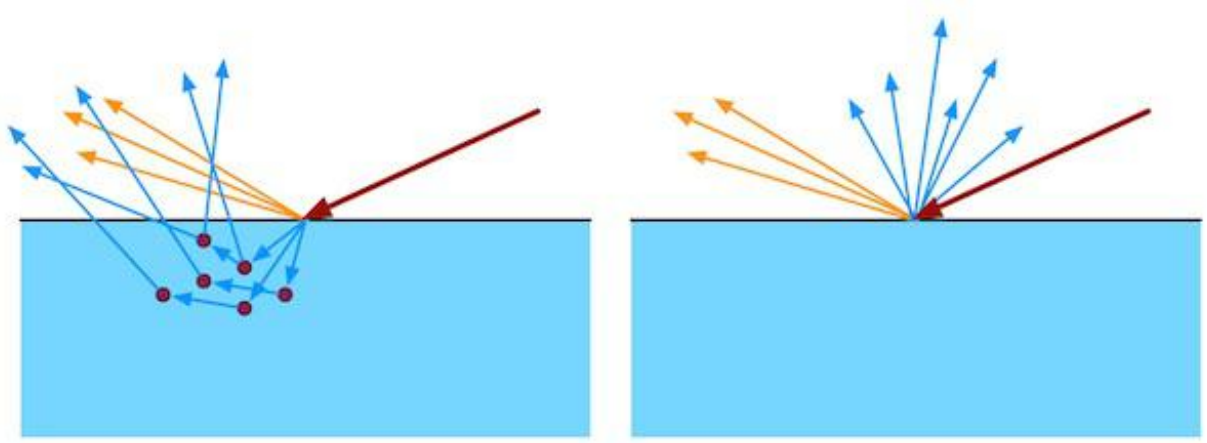


FIG 18.4 subsurface scattering. Left: Subsurface scattering light will be emitted from a position different from the point of incidence. If the distance value smaller than the pixel size to be colored, it can be rendered completely finished (right) part. Otherwise, you need to use subsurface scattering rendering

two phenomena described in Figure 18.5 BRDF. Specular reflection portion for description, the diffuse reflection portion for subsurface scattering describe

FIG 18.6 (a) of those m = h microfacet be exactly incident light reflected from l to v, and only this part can microfacet added to the calculation of the BRDF. Facets (b) satisfies a portion (a) of the element l will be in the other direction microfacet blocking, they do not receive light, therefore shadows. Facets (c) satisfies also a part (a) of the element which will beinreflection direction

blockedhis microfacetv,therefore, is not part of the reflected lightbe seen

tousing FIG 18.7 Standard Shader rendering path in front the Pass (PBS using a simplified version of the structure VertexOutputBaseSimple the like instead of the corresponding structure)

18.8 Unity calibration tables provided in FIG. Left: Metal workflow using calibration tables. Right: Specularworkflow using calibration tables

FIG18.9to implement different types of metal materials workflow. The left sphere: metal. Right sphere:plastic material

18.10 FIGused in the Unity 5 physics-based rendering, a scene at different illuminationrendering results

FIG 18.11 Scenein the lighttabpanel:

of FIG18.12 Left closed when all light sources in the scene and the after ambient light intensity is set to 0, using Standard Shader object still lighting effects. Right: Based on the left, the reflection source is set to null, so that the object does not accept any default information reflecting

FIG 18.13parallel light using

18.14FIG left: the Bounce Intensity set to 0, the object is no longer indirect lighting the impact of shadow detail visible in the part of the little huts. Right: Bounce Intensity to 8, shadow detail more clearly

in FIG left 18.15: Unused reflection probe. Right: two reflection probes placed in the scene, note the differences withleft wall shield

FIG18.16probes with each other using a reflection effect of the reflection of

FIG 18.17 Left: rendering results in linear space. Right:rendering results in gamma space

FIG18.18 perceptual transform human eye is more easily dark region, and less sensitive to changes in bright regions

encoding the gamma and the display of FIG 18.19gamma

18.20FIG Left: gamma space rendering results under. Right:rendering results in a linear space

FIGleft 18.21: gamma results in the mixing space. Right: mixing results in the linear space

Chapter 19 Unity 5updatedany

19.1vertex shader is introduced in the panel, click on the button to view FIG Unity generated forfixed line in FIG shader / fragment shader code