UNIVERSITY OF LEEDS

School of Computing: Assessment brief
计算机学院：评估简报

Module title **模块标题**	Foundations of Modelling and Rendering 建模与渲染基础
Module code **模块代码**	COMP5812M COMP5812M
Assignment title **作业标题**	A1.1 Raytracing and A1.2 Rasterisation. A1.1光线追踪和A1.2光栅化。
Assignment type and description **工作分配类型和说明**	Programming assignment. 程序设计任务。
Rationale **基本原理**	You will be implementing the two most common approaches to rendering 3D scenes, and applying most of the foundational content from this module. 您将实现两种最常用的方法来渲染3D场景，并应用本模块中的大部分基础内容。
Word limit and guidance **字数限制和指导**	This is a pass/fail assignment. You will complete it after a successful demo at the lab class. Attendance is encouraged. 这是一个通过/失败的任务。您将在实验课上成功演示后完成它。鼓励出席。
Weighting **加权**	50%
Submission deadline **提交截止日期**	15/12/2023
Submission method **提交方法**	In-person Demo + Q&A + Gradescope. 现场演示+问答+Gradescope。
Feedback provision **反馈规定**	In-person feedback. 亲自反馈。
Learning outcomes assessed **学习成果评估**	LO1, LO2, LO3, LO4, LO5, LO6, LO7, LO8, LO9, LO10 L01、L02、L03、L04、L05、L06、L07、L08、L09、L010
Module lead **模块引线**	Rafael Kuffner dos Anjos 拉斐尔·库夫纳·多斯安若斯
Other Staff contact **其他员工联系人**	

1. Assignment guidance
作业指导

This assignment is divided into two tasks, the Raytracing and Rasterisation components. Please see the individual description of tasks for each one.
该任务分为两个任务，即光线追踪和光栅化组件。请参阅每个任务的单独说明。

2. Assessment tasks:
评估任务：

A1.1 – Raytracing
A1. 1–光线追踪

This is the first of the two exercises that need to be completed as part of your first assignment. The starting point for this assignment is the end of the Raytracing exercise which is a fully guided tutorial. If you haven't so, please do it now. The final goal is to have a working raytracer with phong shading, hard shadows, and mirrors implemented. This will be the output of correctly implementing the tasks described ahead
这是第一份作业中需要完成的两个练习中的第一个。此作业的起点是光线追踪练习的结尾，这是一个完全有指导的教程。如果你还没有，请现在就做。最终目标是实现一个带有Phong明暗处理、硬阴影和镜像的工作光线跟踪器。这将是正确执行前面描述的任务的输出

Task 1: Transformations
任务1:转换
Your first task will be finishing the code that will perform the transformations. Implement a function that should return your modelview Matrix on the Scene class. Here is a starting point.
您的第一个任务是完成将执行转换的代码。实现一个函数，该函数应在Scene类上返回您的ModelView矩阵。这是一个起点。

```
Matrix4 Scene::getModelview()
Matrix4场景：:GetModelView（）。
{
    Matrix4 result; result.SetIdentity();
    矩阵4结果；result.setIdentity（）；
    //TODO: Grab all the necessary matrices to
    //TODO：获取所有必需的矩阵以
    // build your modelview. And return from this function.
    //构建您的ModelView。并从该函数返回。
    return result;
    返回结果；
}
```

Then, use this function to transform your vertices and normals inside the updateScene(). Example:
然后，使用该函数变换UpdateScene（）中的顶点与法线。示例：

```
//TODO: apply modelview here, before adding vertex to triangle.
//TODO：将顶点添加到三角形之前，在此处应用ModelView。
t.verts[vertex] = v;
T.Verts[顶点]=V；
```

Finally, add a "Scene" object to our RaytraceRenderWidget. Initialize it in the constructor, and call "updateScene()" when we press the raytrace button.
最后，将"Scene"对象添加到我们的RayTraceRenderWidget。在构造函数中初始化它，并在我们按下光线追踪按钮时调用"UpdateScene（）"。
Does it work: Use the debugger to verify that the matrices are being updated accordingly when you use the interface.It will however be easier to tell in the when you are able to see things.
它是否工作：使用调试器来验证当您使用界面时，矩阵是否相应地更新。然而，当您能够看到东西时，将更容易判断。

Task 2: Casting a ray
任务2:投射射线
Implement a function on RaytraceRenderWidget that given a pixel position, casts a ray towards the scene.
在RayTraceRenderWidget上实现一个函数，给定一个像素位置，向场景投射光线。

```
Ray calculateRay(int pixelx, int pixely,bool perspective);
光线计算器（int pixelx，int pixely，bool perspective）；
```

Implement this function by following the process described on Lecture 4, slides 34-38. Call it inside your main raytracing loop, before calculating a colour for a given pixel.
按照第4讲第34-38张幻灯片中描述的过程实现此功能。在计算给定像素的颜色之前，在主光线跟踪循环中调用它。
Does it work: Use the breakpoints to see the coordinates your rays match the edges of your image plane. Those should be similar to what you have as the Left,Right,Bottom,Top from the "glFrustum" on RenderWidget.cpp. It will however be easier to tell in the next step when you are able to see things.
它是否有效：使用断点来查看光线的坐标是否与图像平面的边缘相匹配。它们应该类似于renderwidget.CPP上的"glFrustum"中的"左"、"右"、"下"和"上"。但是，在下一步中，当您能够看到事物时，将更容易分辨。

Task 3: Geometric Intersections
任务3:几何相交
Calculate geometric intersections between camera rays and triangles in the input model. Use an auxiliary struct "CollisionInfo" to contain the intersection information, and implement the following function (starting point below).
计算相机光线与输入模型中的三角形之间的几何交点。使用辅助结构体collisionInfo包含交集信息，并实现以下功能（起点如下）。

```
//scene.h
e.//scenH
    struct CollisionInfo{
                结构碰撞信息{
          Triangle tri;
                三角形三;
        float t;
        浮子T;
    };

CollisionInfo closestTriangle(Ray r);
CollisionInfo最近三角形 (射线R) ;

//scene.cpp
e.//scenCPP
Scene::CollisionInfo Scene::closestTriangle(Ray r)
场景: :CollisionInfo场景: :ClosestTriangle (光线R)
```

```
{
    //TODO: method to find the closest triangle!
    //TODO: 查找最近三角形的方法!
    Scene::CollisionInfo ci;
    场景：:CollisionInfo CI;
    ci.t = r.origin.x; // this is just so it compiles warning free
    CI.T=R.ORIGIN.X; //这只是为了让它编译警告自由
    return ci;
    返回CI;
}
```

This function should iterate the list of triangles in the current scene, and test for intersection with them. The closest triangle given a
此函数应迭代当前场景中的三角形列表，并测试与它们的交集。给定的最接近的三角形
ray will have the smallest t (See Lecture 3, Slides 35-38).
雷的T最小（见第3讲，幻灯片35-38）。
To calculate intersections, implement a method in your Triangle Class.
要计算交点，请在Triangle类中实现一个方法。

```
float intersect(Ray r);
浮动相交（射线R）；
```

Follow the method described in the slides, and return the calculated t for a given intersection, as we can calculate the point of

```
//add to .h
//添加到.H
Cartesian3 baricentric(Cartesian3 o);
Cartesian3重心（Cartesian30）；
//add to .cpp
//添加到.CPP
Cartesian3 Triangle::baricentric(Cartesian3 o)
笛卡尔3三角形：:重心点（笛卡尔3 o）
{
   //TODO: Input is the intersection between the ray and the triangle.
   //TODO: 输入是射线和三角形之间的交点。
   //o = origin + direction*t;
```

按照幻灯片中描述的方法，返回给定交点的计算T，因为我们可以计算
intersection given a ray's origin, direction, and t. We are only interested in cases where t > 0, as negative t would mean an intersection behind the camera. Use this to your advantage to encode "no collision" cases with a negative t value.
给定射线的原点、方向和t的交点。我们只对t>0的情况感兴趣，因为负t意味着摄像机后面的交叉点。利用这一点，您可以使用负T值对"无冲突"情况进行编码。
To verify if the intersection point with the plane formed by the triangle is inside the triangle (thus, a valid intersection) implement the half plane test. (See Lecture 3, Slides 41-44).
要验证与三角形形成的平面的交点是否在三角形内（因此，有效的交点），请执行半平面测试。（参见第3讲第41-44张幻灯片）。
Does it work: Call closestTriangle inside your main raytracing loop. If t > 0, assign the color white to the pixel. Use the sliders and arcball to verify that your transformations are matching with the openGL renderer. Example outputs can be seen on Minerva.
它是否有效：在主光线跟踪循环中调用ClosestTriangle。如果t>0，则为该像素指定白色。使用滑块和弧形球验证变换是否与OpenGL渲染器匹配。示例输出可以在Minerva上看到。

Task 4: Barycentric Interpolation
任务4:重心插值

Implement barycentric interpolation to obtain the barycentric coordinates at the position of the intersection calculated in the
中计算的交点位置处的重心坐标。

previous task. Add a function to your triangle class that returns the "alpha, beta, gamma" as a Cartesian3. Here is a starting point
上一个任务。向Triangle类中添加一个函数，该函数以Cartesian3的形式返回" alpha，beta，gamma "。这是一个起点。
Does it work: Use this function inside your main loop. Use the barycentric coordinates to calculate the normal vector for that point.
它工作吗：在你的主循环中使用这个函数。使用重心坐标计算该点的法向量。
Then set the color of the output pixel as the following, which should show the value of the normal if we have the checkbox
然后将输出像素的颜色设置如下，如果我们有复选框，它应该显示正常的值
"interpolation" checked:

```
if(renderParameters->interpolationRendering)
IF（渲染参数->插值渲染）
       return Homogeneous4(abs(normOut.x),abs(normOut.y),abs(normOut.z),1);
       返回同质4（ABS（normout.X），ABS（normout.y），ABS（norm out.Z），1）；
```

已检查"插值"：
Example outputs can be seen on Minerva.
示例输出可以在Minerva上看到。

Task 5: Blinn-Phong Shading
任务5:Blinn-Phong着色

Calculate Blinn-Phong shading considering every light present in the scene. Use the lights in the vector that you populated in Exercise #2, on the RenderParameters class. Remember to apply the modelview matrix to them as well, as they are tied to a physical object in the scene and these are subject to the sliders and arcball.
考虑场景中存在的每个灯光计算Blinn-Phong明暗处理。使用您在练习#2中填充的矢量中的灯光（在RenderParameters类中）。记住也要将ModelView矩阵应用于它们，因为它们绑定到场景中的物理对象，并且它们受滑块和弧球的影响。
Implement the code to calculate phong shading in a function of your Triangle class (Lecture 4, Slides 7-19). Parameters should be the light position, light colour, and the intersection barycentric coordinates. Use the material properties for that given triangle to calculate phong.
实现在三角形类的函数中计算Phong阴影的代码（第4讲，幻灯片7-19）。参数应为灯光位置、灯光颜色和交点重心坐标。使用给定三角形的材质属性来计算Phong。

Research and implement quadratic attenuation to mimic what happens on the left side (see exercise 4). Return the final colour as Homogeneous4 and apply it to the pixel.
研究并实施二次衰减，以模拟左侧发生的情况（见练习4）。将最终颜色恢复为同质4，并将其应用于像素。
Does it work: Compare it to the shading that happens on the Minerva examples. Change the mtl file of objects you try to see if your implementation responds properly to it.
它是否有效：将其与密涅瓦示例中出现的阴影进行比较。更改您尝试的对象的MTL文件，以查看您的实现是否正确响应它。

Task 6: Shadow Rays
任务6:阴影光线

Include shadows in your shaded result, also considering every light source in the scene. Follow the instructions on the slides (Lecture 5, slides 4-8).
在着色结果中包括阴影，同时考虑场景中的每个光源。按照幻灯片上的说明进行操作（第5讲，幻灯片4-8）。
Use the "closestTriangle" function you implemented before to perform intersection tests, and the t value to check if there was a valid intersection.
使用之前实现的" closestTriangle "函数来执行交集测试，并使用t值来检查是否存在有效的交集。

Pay attention to shadow acne, correctly displacing the starting point of your shadow ray.
注意暗影粉刺，正确地替换你的暗影光线的起点。

Adjust your phong shading function to only apply ambient and emissive colour if the object is in shadow.
如果对象处于阴影中，调整Phong着色功能以仅应用环境色和发射色。
Does it work: Compare to the Minerva examples again. Move the object around, zooming in, and using different angles, ensuring shadow acne never happens.
它是否有效：再次与密涅瓦的例子进行比较。移动周围的对象，放大，并使用不同的角度，确保阴影痤疮永远不会发生。

Task 7: Impulse Reflection
任务7:冲动反思
Use the material properties to verify if the object is mirrored. If it is, calculate the reflected value which should be combined with
使用材质特性验证对象是否已镜像。如果是，则计算应合并的反射值
the object's colour accordingly (values are 0 to 1). This will require you to change your raytracer into a recursive function. Move out all of your shading code after calculating the ray out of the main raytracing loop into a separate function:

```
Ray r = calculateRay(i,j,!renderParameters->orthoProjection); Homogeneous4 color =
TraceAndShadeWithRay(r,N_BOUNCES,1.0f);
光线R=calculateRay (I, J, ! renderParameters->正交投影) ; 均质4颜色=TraceandShadeWithRay (R, N_反弹, 1.0f) ;
//... applying to pixel, gamma correction etc
//...应用于像素、伽玛校正等
```

对象的颜色相应（值为0到1）。这将要求您将光线跟踪器更改为递归函数。在计算主光线跟踪循环中的光线后，将所有着色代码移到单独的函数中：

Where N_BOUNCES is defined in your .h file with the maximum number of recursions allowed in your raytracer.
其中，n_反弹是在.H文件中使用光线跟踪器中允许的最大递归数定义的。
An object is reflective if the mirror property in the material is different than 0. If that's the case, you should find the color at the end of this reflection. Implement a "reflectRay" function that reflects a ray according to a surface normal. `reflectRay(r,normal,hitPoint);`
如果材质中的镜像属性不同于0，则对象是反射的。如果是这种情况，则应在此反射的末尾找到颜色。实现根据曲面法线反射光线的"反射盘"功能。反射盘（R，正常，命中点）；

And recursively call TraceAndShadeWithRay, as described in the slides (Lecture 5, slides 10-13). In the material model we are using
并递归调用TraceAndShadeWithRay，如幻灯片中所述（第5讲，幻灯片10-13）。在我们使用的材料模型中
in this assignment, the "mirror" property in the material file is a floating-point value meaning how much of the total energy should be from the reflected ray. Use this to linearly combine the resulting color of the mirror ray, and the color of the current surface, making sure that no energy is lost, or added to the system.
在此指定中，材质文件中的"镜像"属性是一个浮点值，表示总能量中应有多少来自反射光线。使用该选项可以线性组合镜像光线的结果颜色和当前曲面的颜色，确保没有能量丢失或添加到系统中。
Does it work: Again, check examples on Minerva to compare. Change the material properties to verify that when mirror = 1.0, all you see is a reflection. When mirror=0.0 you should not see any reflection. Anything else should be a combination of colours from phong shading and the reflection.
它是否有效：再次，检查Minerva上的示例以进行比较。更改材质属性以验证当"镜像"（Mirror）=1.0时，所看到的全部是反射。当"镜像"（Mirror）=0.0时，您应该看不到任何反射。其他任何东西都应该是Phong阴影和反射的颜色组合。

A1.2-Rasterisation
A1.2-光栅化

This is the second of the two exercises that need to be completed as part of your first assignment. The goal is to have a working terrain renderer using rasterization, applying phong shading, using materials stored in textures, normal mapping, and a smooth transition between materials according to the height of the terrain. This will be the output of correctly implementing the tasks described ahead.
这是第一次作业中需要完成的两个练习中的第二个。目标是根据地形的高度，使用光栅化、应用Phong着色、使用存储在纹理中的材质、法线贴图和材质之间的平滑过渡来实现工作地形渲染器。这将是正确执行前面描述的任务的输出。

Task 1: Terrain displacement map
任务1:地形位移图
You are provided with a heightmap texture (mountains_height.bmp). This textures encodes what is the height each pixel should have. We are going to use the UV coordinates of the terrain created in the tutorial to map this texture to it, and use it to set the vertex to the correct height in the vertex shader, given a certain scale constant.
将为您提供"高度贴图"（HeightMap）纹理（"山脉"（Mountains）_height.BMP）。该纹理对每个像素应具有的高度进行编码。我们将使用教程中创建的地形的UV坐标将此纹理映射到它，并使用它在顶点着色器中将顶点设置为正确的高度，给定一定的比例常数。

For each pixel in the heighmap, you have a 24 bit integer encoded in the red, green, blue channels of your texture. As an example,
对于高度图中的每个像素，在纹理的红、绿、蓝通道中都有一个24位整数编码。作为一个例子，

let's say a pixel has the height 12345678 This will be encoded as such in RGB values according to the significance of that byte:
假设像素的高度为12345678，这将根据该字节的重要性以RGB值进行编码：

R = 10111100 = 188 = 0.73725	G = 01100001 = 97 = 0.38039	B = 01001110 = 78 = 0.30588

Meaning you can reconstruct the original value using shift operations! 12320768 + 24832 + 78 = 12345678
这意味着您可以使用移位操作重新构建原始值。12320768 + 24832 + 78 = 12345678

Add the required code to load this texture (similarly to what happens in the tutorial) paying attention to what is the sampling mode you are going to use (Lecture 9,
添加所需的代码以加载此纹理（，类似于教程）中发生的情况，请注意您将使用的采样模式（第9讲。

Does it work: You should be seeing mountains! Check the images on Minerva to see if it matches.
它有用吗：你应该看到山！检查密涅瓦上的图像，看看是否匹配。

Task 2: Terrain normals
任务2:地形法线
Calculate a normal per each vertex. This can typically be done by calculating derivatives along both tangents of the plane. There is a simple way of estimating it in our particular case. We can calculate the correct normal vectors by sampling the values around the current pixel and

estimating their differences. Considering we are reading the pixel at position 4, this is one method to calculate it.
计算每个顶点的法线。这通常可以通过计算沿平面的两条切线的导数来完成。在我们的特殊情况下，有一种简单的方法来估计它。我们可以通过对当前像素周围的值进行采样并估计它们的差异来计算正确的法向量。考虑到我们正在读取位置4的像素，这是计算它的一种方法。

1	4	7
2	5	8

Nx = differences between 0-6, 1-7, 2-8 Ny = fixed value.
NX=0-6，1-7，2-8之间的差值NY=固定值。
Nz = differences between 0-2, 3-5, 6-8
NZ=0-2，3-5，6-8之间的差值

Where you can give a higher weight the central differences (1,7,3,5). You are free to research and implement an alternative version if you provide a source for it and can explain it clearly.
您可以为中心差异（1，7，3，5）赋予更高的权重。你可以自由地研究和实现一个替代版本，如果你提供了它的来源，并能清楚地解释它。
Implement this method in your vertex shader, and change the fragment shader to instead of outputting a colour from the texture, to output the normal vector as a colour, such as: vec3(abs(nx),abs(ny),abs(nz)). For that, change the outputs of the vertex shader, and inputs of the fragment shader so the value can go through.
在顶点着色器中实现此方法，并将片段着色器更改为不从纹理输出颜色，而是将法向量输出为颜色，例如：vec3（ABS（NX），ABS（NY），ABS（NZ））。为此，请更改顶点着色器的输出和片段着色器的输入，以便值可以通过。
Does it work: You should see varying colours around the cliffs, and they should keep the same value as you move the camera around.
它工作吗：你应该看到悬崖周围不同的颜色，并且当你移动相机时，它们应该保持相同的值。

Task 3: Terrain shading with texture materials
任务3:使用纹理材质的地形明暗处理
Implement Blinn-Phong shading in your terrain, using the normal vector calculated in the previous step. This should be implemented in your fragment shader. Perform this calculation in the View Coordinate System (See transformations lecture). This will require you passing more matrices as uniforms, and more variables from your vertex to your fragment shader. Set a new uniform with a directional light, which you will use in your calculations.
使用上一步中计算的法线矢量，在地形中实现Blinn-Phong着色。这应该在片段着色器中实现。在视图坐标系中执行此计算（请参阅“转换”讲座）。这将需要你传递更多的矩阵作为制服，更多的变量从你的顶点到你的片段着色器。使用平行光设置新的制服，您将在计算中使用。

```
glm::vec3 lightPos = glm::vec3(0, -0.5, -0.5);
GLM: :VEC3光轴=GLM: :VEC3 (0, -0.5, -0.5) ;
```

Remember that when using a directional light, the light vector is always the same!
请记住，使用平行光时，光向量始终相同！
I have provided a Diffuse and a Specular texture for each one of the different terrains. Continue using the rock and sample these textures to correctly set the materials. This will require adding more textures to the “load textures” method and passing them as uniforms. Be aware of the Filtering mode!
我已经为每一个不同的地形提供了漫反射和镜面反射纹理。继续使用岩石并对这些纹理进行采样，以正确设置材质。这将需要添加更多的纹理到“加载纹理”方法中，并将它们作为统一传递。请注意过滤模式！
Implement tiling in how the textures are sampled, so they are at the right scale. This can be easily achieved by multiplying the UV coordinates by a constant and setting the boundary behaviour properly (GL_TEXTURE_WRAP_S and T).
在纹理采样的方式中实现平铺，以便它们处于正确的比例。这可以通过将UV坐标乘以常量并正确设置边界行为（GL_纹理_包裹_S和T）来轻松实现。
Does it work: You should see parts of the rock texture being shiny, and others not. Check the example on Minerva for details.
它起作用吗：你应该看到岩石纹理的某些部分是有光泽的，而其他部分则没有。有关详细信息，请查看Minerva上的示例。

Task 4: Terrain detailing with normal mapping
任务4:使用法线贴图细化地形
Use the normal map texture provided to apply normal mapping to your terrain. Using the normal vectors you have, and the UV coordinates, calculate the tangent vectors necessary for implementing normal mapping (See the advanced texturing lecture for the method).
使用提供的法线贴图纹理将法线贴图应用于地形。使用已有的法向量和UV坐标，计算实现法线贴图所需的切向量（有关方法，请参见“高级纹理”讲座）。
You should implement this in your vertex shader. Remember that you can calculate the UV coordinates based on the number of
你应该在顶点着色器中实现这一点。请记住，您可以根据
points you have in your mesh. You don’t necessarily need to pass them as arguments. Remember to do Graham-Schmidt process to make sure your basis is orthogonal. Implement the actual normal mapping then in your fragment shader.
网格中的点。您不一定需要将它们作为参数传递。记住做格雷厄姆-施密特过程，以确保你的基础是正交的。实现实际的法线映射，然后在你的片段着色器。
Does it work: Your shading should be a lot more detailed now, even without increasing the resolution of the mesh.
它工作吗：你的着色现在应该更详细，即使没有增加网格的分辨率。

Task 5: Terrain material interpolation
任务5:地形材质插值
You have a texture of grass, rock, and snow. Make it so the fragment shader can choose which texture to apply according to the height of what it is shading. There should be a smooth transition between different textures being used. Pass the height as a parameter between the vertex and the fragment shader, and use it to mix the materials from the different textures.
你有草、岩石和雪的纹理。使片段着色器可以根据着色的高度选择要应用的纹理。使用的不同纹理之间应该有平滑的过渡。将高度作为参数在顶点和片段着色器之间传递，并使用它来混合来自不同纹理的材质。
Does it work: Transitions should be smooth! See example on minerva.
它起作用吗：过渡应该是平滑的！见密涅瓦的例子。

Task 6: Alternative rendering modes and useability
任务6:替代渲染模式和可用性
As a final task, I would like you to implement some functionalities in the renderer to allow some implementation of interaction in the C++ side.
作为最后一项任务，我希望您在渲染器中实现一些功能，以允许在C++端实现一些交互。

- Implement rendering using wireframe that is activated while you press space. Research the function “glPolygonMode” to
- 使用按空格键时激活的线框实现渲染。研究GLPolygonMode函数
  accomplish it.
  完成它。
- Recompile your shaders when you press R. That would allow you to change the shader and recompile them in real time to see the changes in effect.
- 当您按R键时，重新编译您的着色器。这将允许您更改着色器并实时重新编译它们，以查看更改的效果。

- Allow the WASD keys to rotate your directional light.
- 允许WASD键旋转你的平行光。
- Use the T and G keys to control the scale of the terrain.
- 使用T键和G键控制地形的比例。

Does it work: All the buttons should be functional!
它的工作：所有的按钮应该是功能！

3. General guidance and study support
一般指导和研究支持

Consult the Minerva page to obtain the slides for this module, and references in the reading list. Your main source of support for this assignment is the laboratory sessions.
查阅Minerva页面，获取本模块的幻灯片和阅读列表中的参考资料。本作业的主要支持来源是实验课。

4. Assessment criteria and marking process
评估标准和评分流程

Marking will happen during the laboratory classes, thus, the deadline for completing this exercise is the last scheduled laboratory class. However, it is strongly encouraged that you take an iterative approach to this, and work your way through the exercises as
评分将在实验课期间进行，因此，完成此练习的截止日期是最后一节预定的实验课。但是，强烈建议您对此采取迭代方法，并按照您的方式完成练习。
quickly and regularly as possible. There are no “gotchas” in these exercises that we cannot support you with. So: start working on it,
尽可能快速和有规律地。在这些练习中没有我们不能支持你的“陷阱”。所以：开始工作吧，
as soon as you have a problem or don’t know how to progress, ask for support during the labs, and we will help you continue. Demo: When you believe you have completed all the exercises for either rasterization or raytracing, inform the module team, and we will verify that you have completed everything to specification by looking at your prototype running. If there are any problems, we will inform you so you can continue working on it and fix them.
一旦您遇到问题或不知道如何进行，请在实验期间寻求支持，我们将帮助您继续。演示：当您认为您已经完成了光栅化或光线追踪的所有练习时，请通知模块团队，我们将通过查看您运行的原型来验证您是否已经完成了规范中的所有内容。如果有任何问题，我们将通知您，以便您可以继续工作并解决它们。
Q&A: Then we will ask you a few questions about your code that may include explanation about what you did, theoretical questions about the material you had to use to implement it, and requests to change some functionality. The goal of this Q&A session is to verify that you understand the code thoroughly. If you fail to respond to any questions, we will stop the Q&A and resume when you are able to respond correctly. Expect questions related to any one of the implementation tasks, and also to the guided tasks in the exercise.
问与答：然后我们会问你一些关于你的代码的问题，可能包括关于你做了什么的解释，关于你必须用来实现它的材料的理论问题，以及改变一些功能的请求。此问答环节的目标是验证您是否完全理解代码。如果您未能回答任何问题，我们将停止问答，并在您能够正确回答时继续。预计会出现与任何一项实施任务相关的问题，以及与练习中的指导任务相关的问题。

5. Presentation and referencing
演示和引用

Questions will be asked and answered in English. Comments on your code must be written in English or they will be ignored.
问题将用英语提问和回答。代码上的注释必须用英语编写，否则将被忽略。

6. Submission requirements
提交要求

Gradescope Submission: After each demo and Q&A steps are completed, you will be given a “digital receipt” that you have been
Gradescope提交：在每个演示和问答步骤完成后，您将获得一张“数字收据”
assessed.
已评估。
A1.1: Prepare a zip folder with the “Raytracing” project folder and the receipt file, but without the “objects” folder.
A1.1:准备一个zip文件夹，其中包含“光线追踪”项目文件夹和收据文件，但不包含“对象”文件夹。
A1.2: . Prepare a zip folder with the receipt and the following files: main.cpp, Basic.vert, Texture.frag.
答案1.2:。c.准备一个包含收据和以下文件的zip文件夹：main.CPP、basie.vert、
texturfrag。
Please do not submit anything else, as the total size of the project will be too large.
请不要提交任何其他内容，因为该项目的总规模将过于庞大。

Submit both zip files to Gradescope as a single submission. I would encourage you to submit as soon as you are done for “backup”,
将两个ZIP文件作为单个提交提交到Gradescope。我鼓励您在完成“备份”后尽快提交，
and update when you finish both exercises.
并在完成两个练习时进行更新。

7. Academic misconduct and plagiarism
学术不端与剽窃

All of the code submitted will be verified using plagiarism detection software. You are not allowed to submit code that was not written by you. In this assignment, it is not allowed to use any external sources for the code you submitted, even with references. All the submitted code must be written from the start by you.
提交的所有代码都将使用剽窃检测软件进行验证。不允许您提交不是由您编写的代码。在此作业中，不允许对您提交的代码使用任何外部源，即使有引用。所有提交的代码必须从一开始就由您编写。

The Q&A sessions will be also used to verify that you wrote the code yourself, and that you are familiar with all aspects of it. If any member of staff suspects of plagiarism at this point, you will be warned at this point and your submission will be closely investigated after submission to Gradescope.
问答环节还将用于验证您是否自己编写了代码，以及您是否熟悉代码的各个方面。如果任何工作人员在这一点上怀疑剽窃，您将在这一点上受到警告，并且您的提交将在提交给Gradescope后受到密切调查。

8. Assessment/ marking criteria grid
评估/评分标准网格

50 points: Everything works, Q&A completed successfully.

50分：一切正常，问答成功完成。