# Test #2

**Due** Mar 26 at 11:59pm **Points** 100 **Questions** 40 **Available** Mar 22 at 12:01pm - Mar 26 at 11:59pm
**Time Limit** 60 Minutes

## Instructions

**Congratulations on getting to Finals Week! It has been a real pleasure having all of you in CS 457/557 this quarter. Thanks for being here!**

### Attempt History

| | Attempt | Time | Score |
|---|---|---|---|
| **LATEST** | Attempt 1 | 39 minutes | 97.5 out of 100 |

ⓘ Correct answers will be available on Mar 27 at 12:01am.

Score for this quiz: **97.5** out of 100
Submitted Mar 26 at 9:14pm
This attempt took 39 minutes.

**Question 1** **2.5 / 2.5 pts**

**If you want to *transform (enlarge, rotate, etc.) an image*, the transformation you need to perform on the *image's s and t coordinates* is:**

- ○ The transverse of the desired image transform
- ◉ The inverse of the desired image transform
- ○ The same as the desired image transform
- ○ The transpose of the desired image transform

**Question 2** **2.5 / 2.5 pts**

**In the Geometry Shader Project, what does this code do?**

```
f *= 10.;
int fi = int( f + 0.5 );
f = float( fi ) / 10.;
```

- ○ Rounds the number f to the nearest integer
- ◉ Rounds the number f to the nearest multiple of 0.1
- ○ Rounds the number f to the nearest multiple of 10.
- ○ Rounds the number f to the next lower integer

## Question 3

**2.5 / 2.5 pts**

**Line Integral Convolution is a way to perform:**

- ○ Vector flow visualization by drawing a series of streamlines using line strips
- ○ Scalar flow visualization by smearing the flow field
- ○ Scalar flow visualization by drawing a series of streamlines using line strips
- ◉ Vector flow visualization by smearing the flow field

## Question 4

**2.5 / 2.5 pts**

**To display a shadow, you:**

- ○ Paint the shadow area a shade of gray
- ○ Paint just the diffuse and skip the ambient and specular
- ○ Paint the shadow area black
- ◉ Paint just the ambient and skip the diffuse and specular

## Question 5

**2.5 / 2.5 pts**

**When a Compute Shader extracts information from an array of all Work Items (such as in the particle system program), the array index comes from:**

- ◉ gl_GlobalInvocationID
- ○ gl_LocalInvocationID
- ○ gl_WorkGroupID
- ○ gk_WorkGroupSize

## Question 6

**2.5 / 2.5 pts**

**When using the fragment shader to work with images, you read the image in as a texture whose dimensions are ResS and ResT. From the (s,t) location where you are right now, how do you look at one texel to the right?**

- ○ Add (1.,1.) to (s,t)
- ○ Add( 1.,0.) to (s,t)

- Add (1./ResS,1./ResT) to (s,t)
- [x] Add (1./ResS,0.) to (s,t)

## Question 7

**2.5 / 2.5 pts**

**In the silhouette shader, why are *triangles_adjacency* used as the input topology?**

- Because it is the only triangle-ish toplogy that geometry shaders are allowed to use
- Because we need to look at the s and t coordinates of the adjacent triangles
- [x] Because we need to look at the normals of the adjacent triangles

## Question 8

**2.5 / 2.5 pts**

**What is the one thing that a Tessellation Shader can do that *no other shader* can do?**

- Draw a Bezier curve
- Create more detail
- [x] Apply built-in patterns
- Change toplogy

## Question 9

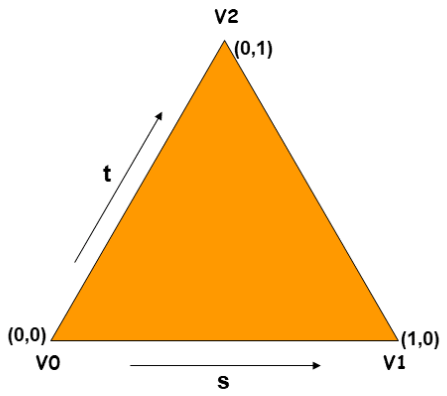
**2.5 / 2.5 pts**

**Which of these is *not* one of the 5 Geometry Shader legal *input* topologies?**

- GL_POINTS
- GL_TRIANGLES
- GL_LINES_WITH_ADJACENCY
- [x] GL_QUADS

## Question 10

**2.5 / 2.5 pts**




**Given this diagram, what is the triangle interpolation equation?**

- [x] V = V0 + s*(V1-V0) + t*(V2-V0)
- [ ] V = (V1-V0) + s*(V2) + t*(V1)
- [ ] V = V0 + s*(V2-V0) + t*(V1-V0)
- [ ] V = (V1-V0) + s*(V1) + t*(V2)

## Question 11

**2.5 / 2.5 pts**

**The special thing about Shader Storage Buffer Objects, compared with other OpenGL buffer objects is:**

- [ ] They can hold floats as well as ints
- [x] Your shaders can both read from them and write to them
- [ ] They can hold doubles as well as floats

## Question 12

**2.5 / 2.5 pts**

**Surface Local Coordinates are used to:**

- [ ] Apply bump-mapping to planar surfaces
- [ ] Apply bump-mapping to create ripple effects
- [ ] Apply bump-mapping to create the appearance of terrains
- [x] Apply bump-mapping to non-planar surfaces

## Question 13

**2.5 / 2.5 pts**

**What is the un-mask image (i.e., what you don't want) for Saturation?**

- ○ All gray
- ○ All white
- ◉ Luminance/Grayscale
- ○ All Black

## Question 14

**2.5 / 2.5 pts**

**How does a Jitter Cloud differ from a Point Cloud?**

- ○ The Jitter Cloud adds some points to the Point Cloud to create more detail
- ◉ The Jitter Cloud has its points randomly moved to eliminate distracting artifact patterns
- ○ The Jitter Cloud removes some points from the Point Cloud to de-clutter the scene
- ○ There is no difference – they are two phrases that describe the same thing

## Question 15

**2.5 / 2.5 pts**

**A scene depicting a street oil slick has rainbow colors because:**

- ◉ At each fragment, there will be one wavelength (color) that reinforces and thus is visible
- ○ At each fragment, there will be one wavelength (color) that cancels and thus is visible
- ○ At each fragment, you look for a light-fragment-eye angle between 40 and 42 degrees

## Question 16

**2.5 / 2.5 pts**

**In RenderMan, what is a *microfacet*?**

- ○ A subset of the rendering process
- ○ A sub-pixel, usually smaller than a pixel
- ○ It's the same as a polygon, just a different word for it
- ◉ A sub-polygon, usually smaller than a pixel

## Question 17

**2.5 / 2.5 pts**

**What is one Point Cloud problem that a Jitter Cloud solves?**

- [x] The row-of-corn problem
- [ ] Lighting-fighting
- [ ] Texturing
- [ ] Z-fighting

## Question 18

**2.5 / 2.5 pts**

**What is the un-mask image (i.e., what you don't want) for Contrast?**

- [ ] Luminance/Grayscale
- [ ] All white
- [ ] All black
- [x] All Gray

## Question 19

**2.5 / 2.5 pts**

**A Tessellation Evaluation Shader (TES) is given a (u,v,w) and produces:**

- [ ] An (s,t,p)
- [x] An (x,y,z)
- [ ] An (s,t)
- [ ] An s

## Question 20

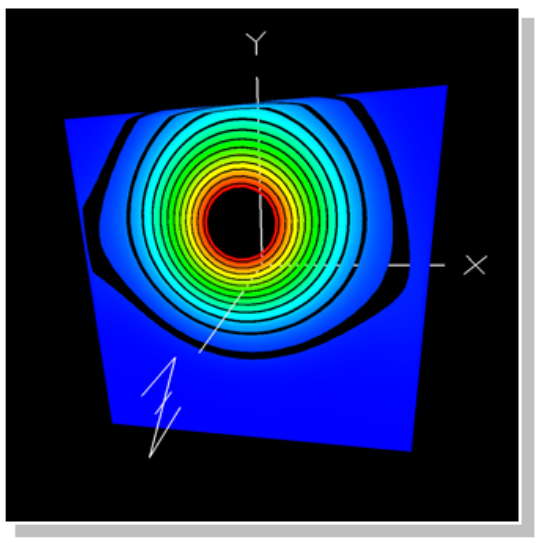
**2.5 / 2.5 pts**

**The goals of RenderMan vs. the goals of GLSL are:**

- [ ] The same -- both emphasize rendering quality above all else
- [ ] GLSL emphasizes rendering quality, RenderMan emphasizes rendering speed
- [x] GLSL emphasizes rendering speed, RenderMan emphasizes rendering quality
- [ ] The same -- both emphasize rendering speed above all else

## Question 21

2.5 / 2.5 pts




**When using the fragment shader to color the temperature-data cutting plane, you can add the see-through gapped contours, like is being shown here, by:**

- (●) Checking how close the temperature is to a multiple of 10., then discard
- ( ) Checking how close the color is to black, then set alpha=0
- ( ) Checking how close the color is to black, then discard
- ( ) Checking how close the temperature is to a multiple of 10., then set alpha=0.

## Question 22

2.5 / 2.5 pts

**Which of these is *not* a way that an OpenGL-GLSL shader differs from a Vulkan-GLSL shader? (I.e., which of these is false?)**

- ( ) A Vulkan GLSL shader must give a set-number for all uniform variables
- (●) An OpenGL GLSL shader compiler has a built-in #define called OPENGL
- ( ) A Vulkan GLSL shader must be pre-compiled with an external compiler
- ( ) A Vulkan GLSL shader compiler has a built-in #define called VULKAN

## Question 23

2.5 / 2.5 pts

**A Transfer Function:**

- ○ Transfers data from the GPU back to the CPU
- ◉ Turns a scalar data value into its appropriate red, green, blue, and alpha
- ○ Transfers data from the CPU over to the GPU
- ○ Transfers a function from your C/C++ code to your GLSL program

## Question 24

**2.5 / 2.5 pts**

**Image Un-masking is a recogntion of the fact that:**

- ◉ It is sometimes easier to ask for what you don't want than to ask for what you do want.
- ○ It is usually easier to derive the mage maipulation equations from scratch
- ○ It is sometimes better to filter an image by and'ing with a bit-mask
- ○ It is sometimes better to filter an image by or'ing with a bit-mask

## Question 25

**2.5 / 2.5 pts**

**When we discussed Cube-mapping with refraction, we said that you could not refract out the back of an object. Yet, when we talked about lenses, we found out that we *could* refract out the back. How did we accomplish this bit of trickery?**

- ○ we did it as a two-pass render
- ○ We rendered it with very tiny "micropolygons" instead of our normally-sized polygons
- ○ We had to use two separately-written shaders
- ◉ We knew the equations for the lens surfaces

## Question 26

**2.5 / 2.5 pts**

**In modeling wave motion, the individual vertices move:**

- ○ Up and down
- ○ Up, down, left, and right
- ◉ In circles
- ○ Left and right

## Question 27

**2.5 / 2.5 pts**

**What is the purpose of the *second* vertex+fragment shader in the shadow operation?**

- ◉ It uses the depth map to decide if a given fragment is blocked from seeing the light source
- ○ It generates the depth map from the eye's point of view
- ○ It animates the objects in the scene
- ○ It picks the color to paint the shadow

## Question 28

**2.5 / 2.5 pts**

**In using shaders to create a contour plane for visualization, the way to turn a fragment's XYZ in the range (-1.,+1.) into an STP in the range (0.,1.) is:**

- ○ vec3 stp = dot( stp, stp );
- ○ vec3 stp = -1. + ( 2. * xyz );
- ◉ vec3 stp = ( xyz + 1. ) / 2.;
- ○ vec3 stp = xyz;

## Question 29

**2.5 / 2.5 pts**

**The Silhouette Shader works by starting with a triangle and:**

- ◉ Examining adjacent triangles and looking for a sign-change in the z-component of their normals
- ○ Drawing the 3 edges of the triangle with a slightly closer z coordinate to avoid z-fighting
- ○ Checking the curvature within the triangle for a point of inflection
- ○ Checking its own 3 per-vertex normals and looking for a sign-change in their z-component

## Question 30

**2.5 / 2.5 pts**

**Parallax Mapping:**

- (•) Is a form of bump-mapping in that no surface is actually displaced
- ( ) Causes surfaces to actually be displaced, unlike bump-mapping

## Question 31

**2.5 / 2.5 pts**

**An example of an application that *requires* Render-to-Texture operations is:**

- ( ) Drawing a Julia set
- ( ) Applying a tessellation to a triangle shrinking operation
- (•) Applying an image edge-detection to a 3D scene
- ( ) Morphing a cow

## Question 32

**2.5 / 2.5 pts**

**When we want to latch a  set of vertex output values in a Geometry Shader, we call**

- ( ) glEmitVertex( )
- ( ) EndPrimitive( )
- ( ) glEndPrimitive( )
- (•) EmitVertex( )

Incorrect

## Question 33

**0 / 2.5 pts**

**Vulkan Ray-tracing involves:**

- ( ) Using compute shaders
- (•) Using two new types of GLSL shader
- ( ) Using five new types of GLSL shaders
- ( ) Using all the pipeline shaders we already know about

## Question 34

2.5 / 2.5 pts

**A Compute Shader differs from all other OpenGL shaders in that:**

- ○ It cannot call the built-in functions, such as reflect( )
- ○ It cannot use 32-bit integers and floats
- ◉ It is not part of the graphics pipeline
- ○ It cannot access any of the uniform variables

## Question 35

2.5 / 2.5 pts

**What does this line of code do (in a 1D compute shader)?**

**uint gid = gl_GlobalInvocationID.x**

- ◉ It tells us where we are in the list of all (global) work items
- ○ It reads one group from the list of all work groups
- ○ It reads one item from the list of all work items
- ○ It tells us where we are in the list of just the local work items

## Question 36

2.5 / 2.5 pts

**What is the purpose of the *first* vertex+fragment shader combination in the shadow operation?**

- ○ It picks the color to paint the shadow
- ◉ It generates the depth map from the eye's point of view
- ○ It animates the objects in the scene
- ○ It uses the depth map to decide if a given fragment is blocked from seeing the light source

## Question 37

2.5 / 2.5 pts

**Vulkan was developed to be:**

- ○ A replacement for Direct3D
- ○ A replacement for OpenGL

- (selected) A more efficient computer graphics API
- A Python equivalent to OpenGL

## Question 38

**2.5 / 2.5 pts**

**What do the Primitive Assembly stages of the graphics pipeline do?**

- (selected) Gather individual vertices into an array before proceeding
- Build a quadrilateral from individual normals
- Guarantee that the texture coordinates correspond to actual texels
- Guarantee that the color values actually fall between 0. and 1.

## Question 39

**2.5 / 2.5 pts**

**What is the one thing that a Geometry Shader can do that *no other shader* can do?**

- Draw a Bezier curve
- Apply built-in patterns
- Create more detail
- (selected) Change toplogy

## Question 40

**2.5 / 2.5 pts**

**Normal-mapping (e.g., the bricks demo) differs from usual Bump-mapping (e.g., the ripples) in that:**

- In Normal-mapping, the surface normal vectors are known only in some places, but not all
- In Normal-mapping, the surface normal vectrs are known everywhere from a cross product
- In Normal-mapping, the surface normal vectors are known everywhere from an equation
- (selected) In Normal-mapping, the surface normal vectors are known everywhere from a texture image

Quiz Score: **97.5** out of 100