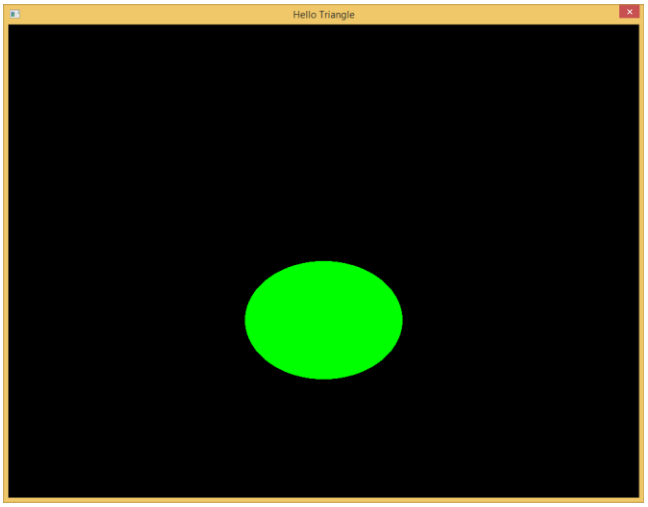
DEBUG WITH VISUAL STUDIO - PRACTICE

Document: DEBUG WITH VISUAL STUDIO.PDF OR DEBUG WITH VISUAL STUDIO.DOCX

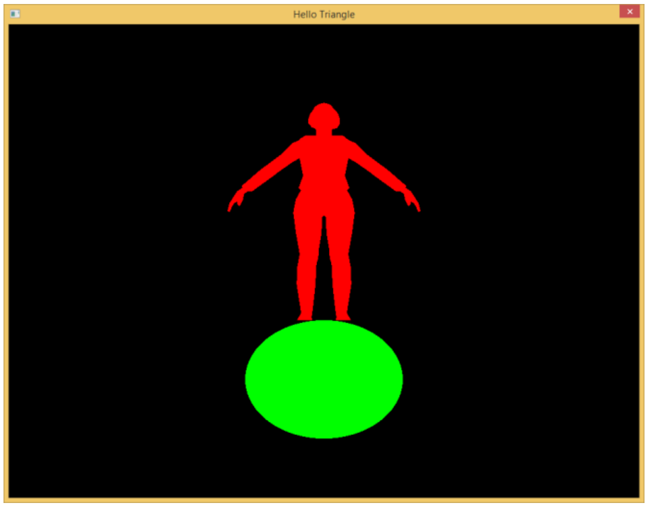
Please read carefully the project ***VSC_Debug_Practice.zip*** and do following requests:

RENDERING

Currently, the project is getting troubles so it's displayed incorrectly:



Debug the project to fix it, the correct rendering is as below:



<u>Then, explain the reason here:</u>

....

CHANGE DATA FILES

Currently, the project is loading the model file named "`Woman1.nfg`", accordingly with "Girl.fs" and "Girl.vs".

Now, please:

Option 1:

Replace rendering "Woman1.nfg" to "**Woman2.nfg**", still with "Girl.vs" and "Girl.fs"

Option 2:

Replace rendering “Woman1.nfg” to “**Croco.nfg**”, accordingly “Girl.vs and Girl.fs” to “**Crodo.vs and Croco.fs**”

Option 3:

Replace rendering “Woman1.nfg” to “**House.nfg**”, accordingly “Girl.vs and Girl.fs” to “**House.vs and House.fs**”

Option 4:

Replace rendering “Woman1.nfg” to “**Marine.nfg**”, accordingly “Girl.vs and Girl.fs” to “**Marine.vs and Marine.fs**”

Option 5:

Replace rendering “Woman1.nfg” to “**bus.nfg**”, accordingly “Girl.vs and Girl.fs” to “**Others.vs and Others.fs**”

Option 6:

Replace rendering “Woman1.nfg” to “**Goliath.nfg**”, accordingly “Girl.vs and Girl.fs” to “**Others.vs and Others.fs**”

Option 7:

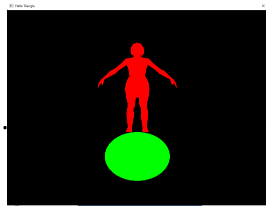
Replace rendering “Woman1.nfg” to “**Radar.nfg**”, accordingly “Girl.vs and Girl.fs” to “**Others.vs and Others.fs**”

Option 8:

Replace rendering “Woman1.nfg” to “**witch.nfg**”, accordingly “Girl.vs and Girl.fs” to “**Witch.vs and Witch.fs**”

Take a picture of rendering result here.

Answer:



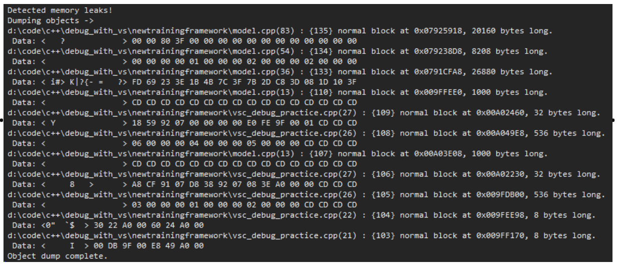
....

MEMORY LEAK DETECT

Request 1

Take a picture of Output Window displaying all leaked memory in which line, which file.

Answer:



```
Detected memory leaks!
Dumping objects ->
d:\code\c++\debug_with_vs\newtrainingframework\model.cpp(83) : {135} normal block at 0x07925918, 20160 bytes long.
 Data: <   ?            > 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00
d:\code\c++\debug_with_vs\newtrainingframework\model.cpp(54) : {134} normal block at 0x079238D8, 8208 bytes long.
 Data: <                > 00 00 00 00 01 00 00 00 02 00 00 00 02 00 00 00
d:\code\c++\debug_with_vs\newtrainingframework\model.cpp(36) : {133} normal block at 0x0791CFA8, 26880 bytes long.
 Data: < i#> K|?{- =   ?> FD 69 23 3E 1B 4B 7C 3F 7B 2D C8 3D 08 1D 10 3F
d:\code\c++\debug_with_vs\newtrainingframework\model.cpp(13) : {110} normal block at 0x009FFEE0, 1000 bytes long.
 Data: <                > CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
d:\code\c++\debug_with_vs\newtrainingframework\vsc_debug_practice.cpp(27) : {109} normal block at 0x00A02460, 32 bytes long.
 Data: < Y              > 18 59 92 07 00 00 00 00 E0 FE 9F 00 01 CD CD CD
d:\code\c++\debug_with_vs\newtrainingframework\vsc_debug_practice.cpp(26) : {108} normal block at 0x00A049E8, 536 bytes long.
 Data: <                > 06 00 00 00 04 00 00 00 05 00 00 00 CD CD CD CD
d:\code\c++\debug_with_vs\newtrainingframework\model.cpp(13) : {107} normal block at 0x00A03E08, 1000 bytes long.
 Data: <                > CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
d:\code\c++\debug_with_vs\newtrainingframework\vsc_debug_practice.cpp(27) : {106} normal block at 0x00A02230, 32 bytes long.
 Data: <    8   >       > A8 CF 91 07 D8 38 92 07 08 3E A0 00 00 CD CD CD
d:\code\c++\debug_with_vs\newtrainingframework\vsc_debug_practice.cpp(26) : {105} normal block at 0x009FDB00, 536 bytes long.
 Data: <                > 03 00 00 00 01 00 00 00 02 00 00 00 CD CD CD CD
d:\code\c++\debug_with_vs\newtrainingframework\vsc_debug_practice.cpp(22) : {104} normal block at 0x009FEE98, 8 bytes long.
 Data: <0"  `$  > 30 22 A0 00 60 24 A0 00
d:\code\c++\debug_with_vs\newtrainingframework\vsc_debug_practice.cpp(21) : {103} normal block at 0x009FF170, 8 bytes long.
 Data: <    I   > 00 DB 9F 00 E8 49 A0 00
Object dump complete.
```

....

Request 2

How many memory bytes leaks exactly?

Answer:

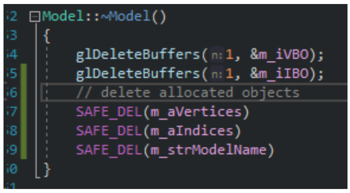
58400 bytes

Request 3

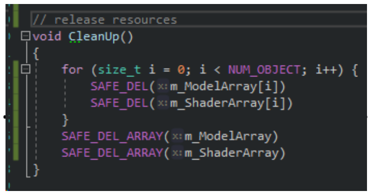
Fix all the memory leaks.

Take picture of coding where you fix memory leak to this file for the answer. Explain/give comments if any too.

Answer:



```
Model::~Model()
{
    glDeleteBuffers(n:1, &m_iVBO);
    glDeleteBuffers(n:1, &m_iIBO);
    // delete allocated objects
    SAFE_DEL(m_aVertices)
    SAFE_DEL(m_aIndices)
    SAFE_DEL(m_strModelName)
}
```



```
// release resources
void CleanUp()
{
    for (size_t i = 0; i < NUM_OBJECT; i++) {
        SAFE_DEL(x:m_ModelArray[i])
        SAFE_DEL(x:m_ShaderArray[i])
    }
    SAFE_DEL_ARRAY(x:m_ModelArray)
    SAFE_DEL_ARRAY(x:m_ShaderArray)
}
```

WATCH WINDOW

Request 1

In method:

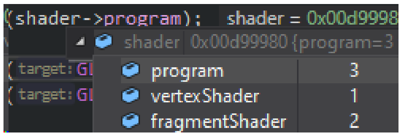
```
void Draw(Shaders* shader, Model* model)
{
        glUseProgram(shader->program);
....
```

Without using any Window (Watch Window, Inmediate Window, ect.), how to know value of **shader->program**?

Take the picture of the answer.

Answer:

hover your mouse on it or print to output window



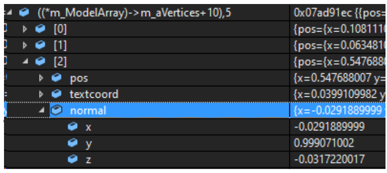
```
(shader->program);   shader = 0x00d9998
  ◢ shader  0x00d99980 {program=3
( target: GI     program          3
( target: GI     vertexShader     1
                 fragmentShader   2
```

Request 2

Take a picture of watch window to see values of pos, textcoord, normal, biNormal and tangent in **m_aVertices** array with below condition:

1. Just view from 10th to 15th index.
2. Highlight value of m_aVertices[12].normal in watch window.

Answer:

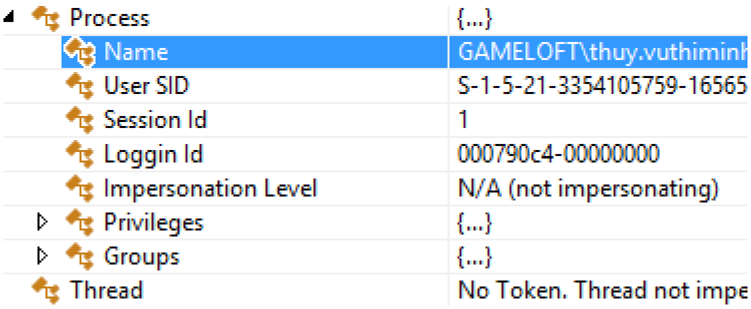


```
◢ ((*m_ModelArray)->m_aVertices+10),5   0x07ad91ec {{pos=
   ▷ [0]                                 {pos={x=0.108111
   ▷ [1]                                 {pos={x=0.063481
   ◢ [2]                                 {pos={x=0.547688
      ▷ pos                              {x=0.547688007 y=
      ▷ textcoord                        {x=0.0399109982 y
      ◢ normal                           {x=-0.0291889999
           x                             -0.0291889999
           y                             0.999071002
           z                             -0.0317220017
```

Request 3

Take a picture of Watch Window to see some values on your application:

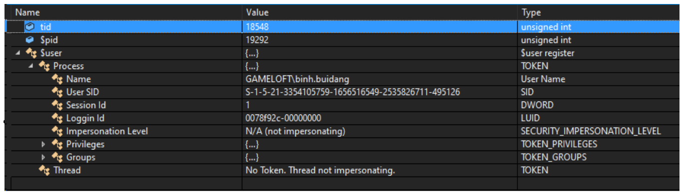
1. Thread ID of the current thread
2. The process ID
3. Information for the account running the program, something like this:



Process	{...}
Name	GAMELOFT\thuy.vuthimint
User SID	S-1-5-21-3354105759-16565
Session Id	1
Loggin Id	000790c4-00000000
Impersonation Level	N/A (not impersonating)
Privileges	{...}
Groups	{...}
Thread	No Token. Thread not impe

4. The command line string that launched the program.

Answer:



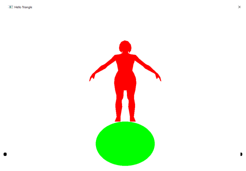
Name	Value	Type
tid	18548	unsigned int
$pid	19292	unsigned int
$user	{...}	$user register
Process	{...}	TOKEN
Name	GAMELOFT\binh.buidang	User Name
User SID	S-1-5-21-3354105759-1656516549-2535826711-495126	SID
Session Id	1	DWORD
Loggin Id	0078f92c-00000000	LUID
Impersonation Level	N/A (not impersonating)	SECURITY_IMPERSONATION_LEVEL
Privileges	{...}	TOKEN_PRIVILEGES
Groups	{...}	TOKEN_GROUPS
Thread	No Token. Thread not impersonating.	TOKEN

....

Request 4

What if you call `SetScreenColor(1.0f, 1.0f, 1.0f, 1.0f)` in Watch Window, take the picture of result rendering window.

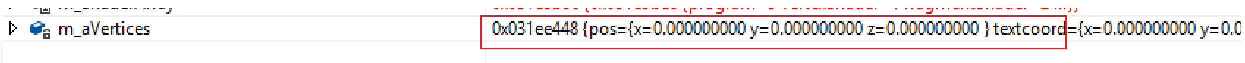
Answer:



....

Request 5

As default, m_aVertices is displayed in Watch Window as current format: “pos={pos_value} normal={normal_value} binormal={binormal_value} tagent={tangent_value} texcoord={textcoord_value}”. See below picture for illustration:

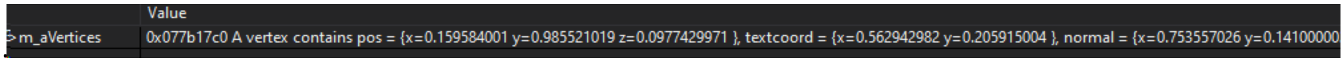


```
m_aVertices    0x031ee448 {pos={x=0.000000000 y=0.000000000 z=0.000000000 } textcoord={x=0.000000000 y=0.0
```

Now, please modify this visualizer type to display exactly as format: “A vertex contains: pos={pos_value}, textcoord={textcoord_value}, normal={normal_value}, binormal={binormal_value} and tangent={tangent_value}”

Then, take the picture of watch window here.

Answer:



```
                Value
m_aVertices     0x077b17c0 A vertex contains pos = {x=0.159584001 y=0.985521019 z=0.0977429971 }, textcoord = {x=0.562942982 y=0.205915004 }, normal = {x=0.753557026 y=0.14100000
```