Qualcomm Developer Network

GaussianFilter

|  |  |  |
| --- | --- | --- |
| **Attribution:** |  | |
| **Email address** | [sunzhen@thundersoft.com](mailto:sunzhen@thundersoft.com)  [yangrong0925@thundersoft.com](mailto:yangrong0925@thundersoft.com)  [Kouzw0723@thundersoft.com](mailto:Kouzw0723@thundersoft.com)  [yansq0930@thundersoft.com](mailto:yansq0930@thundersoft.com)  zhangxu0710@thundersoft.com | |
| **Project Title**\* | **GaussianFilter** | |
| **Images**  *Upload up to 5 images of your project*  *Please submit/send the original JPEG/PNG files for all images included in the document* | **GaussianFiltermain window.png**  /home/zx/Pictures/2020-08-20 14-20-17 的屏幕截图.png2020-08-20 14-20-17 的屏幕截图/home/zx/Pictures/2020-09-04 13-52-59 的屏幕截图.png2020-09-04 13-52-59 的屏幕截图  [alt tag: “When the application starts, you will see a image model, which will change over time.”] | |
| **Description**\*  *High level description of the project* ***(75 words or less)*** | Develop a demo application using the Adreno GPU SDK to display Smoothing filtermodels. | |
| **Objective**   * *What inspired you to create this project?* * *What is your desired outcome?* | Qualcomm Adreno GPU has powerful image filter performance. I hope to do a simulated wave application to demonstrate this performance, and use Snapdragon Profiler software to analyze CPU and GPU performance usage. | |
| **Materials Required / Parts List / Tools** | Part Name | Link to purchase |
| Adreno GPU SDK | <https://developer.qualcomm.com/software/adreno-gpu-sdk> |
| Android NDK | https://  developer.android.google.cn/ndk/downloads/ |
| Apache Ant | <https://ant.apache.org/manualdownload.cgi> |
| Snapdragon Profiler | <https://developer.qualcomm.com/software/snapdragon-profiler> |
| **Source Code / Source Examples / Application Executable**  *Link to open source / shareable code repository* | Description | Link |
| [Source Code](https://github.com/canyudeguang/Home_Automation) | https://github.com/ThunderSoft-XA/Adreno-GPU-SDK-PROJECT.git |
| **Additional Resources**  *List related links or resources such as websites, videos, presentations, or other materials* | Resource Title | Link or File Name (and provide file) |
| Video Links |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Build / Assembly Instructions** | Parts used Below are the items used in this project.   |  |  | | --- | --- | | **Phone**  **/home/zx/Pictures/SDM865.jpegSDM865** | **Snapdragon Profiler**  Snap |   [alt tag: “**Parts used for developing.** ”]   1. Android device with Snapdragon SDM845 processors installed the apk compiled by Android NDK, used to run apk and see image filter display effect. 2. PC with Ubuntu 18.04. 3. Type-C data cable. 4. Adreno GPU SDK v5.0. 5. Android NDK r17b. 6. Apache Ant 1.9.14. 7. Snapdragon Profiler, a tool used to view apk performance indicators in real time.   Deploying the project   1. Download Adreno GPU SDK v5.0 from <https://developer.qualcomm.com/software/adreno-gpu-sdk>, and install it to PC. 2. Download Android NDK r17b and install it to PC. 3. Download Apache Ant 1.9.14 and install it to PC. 4. Download Snapdragon Profiler from <https://developer.qualcomm.com/software/snapdragon-profiler>, and install it to PC. 5. Realize the wave shader. 6. Call the wave shader in the APK. 7. Display FPS in the APK. 8. Realize required functions step by step. 9. Choose android platform and connect android device, then complie the project, then install the APK. 10. Open it and you’ll enjoy. 11. if you care about some performance indicators, you can use Snapdragon Profiler to view it. 12. If no problem, upload code to Github. | |
| **Project Walkthrough** | How does it work?  **1、How to realize the smoke shader？**  In order to enable OpenGL to achieve wave drawing, we must first create a shader.  **Vertex Shader：**  const GLchar \* vs\_common = STRINGIZE(  uniform mat4 MVP;  layout (location = 0) in vec4 Position;  layout (location = 1) in vec4 Color;  layout (location = 2) in vec2 UV;  out vec4 FragColor;  out vec2 TexCoord;  void main(void)  {  FragColor = Color;  TexCoord = UV;  gl\_Position = MVP \* Position;  }  );  **Fragment Shader：**  const GLchar \* fs\_frag\_tex = STRINGIZE(  uniform sampler2D Texture;  in vec4 FragColor;  in vec2 TexCoord;  layout (location = 0) out vec4 Color;  void main(void)  {  vec4 texel = texture(Texture, TexCoord);  if (texel.a < 0.5)  {  discard;  }  else  {  Color = texel \* FragColor;  }  }  );  const GLchar \* fs\_tex\_filter = STRINGIZE(  uniform sampler2D Texture;  uniform uint Filter;  in vec4 FragColor;  in vec2 TexCoord;  layout (location = 0) out vec4 Color;  vec4 texel;  void clamp\_vec4(void)  {  for (int i = 0; i < 4; i++)  {  texel[i] = max(0.0, min(1.0, texel[i]));  }  }  void sort2(inout vec4 a0, inout vec4 a1)  {  vec4 b0 = min(a0, a1);  vec4 b1 = max(a0, a1);  a0 = b0;  a1 = b1;  }  void sort5(inout vec4 a0, inout vec4 a1, inout vec4 a2, inout vec4 a3, inout vec4 a4)  {  sort2(a0, a1);  sort2(a3, a4);  sort2(a0, a2);  sort2(a1, a2);  sort2(a0, a3);  sort2(a2, a3);  sort2(a1, a4);  sort2(a1, a2);  sort2(a3, a4);  }  void f\_7x7(in float coefs[49])  {  vec4 c00 = textureOffset(Texture, TexCoord, ivec2(-3, -3));  vec4 c01 = textureOffset(Texture, TexCoord, ivec2(-2, -3));  vec4 c02 = textureOffset(Texture, TexCoord, ivec2(-1, -3));  vec4 c03 = textureOffset(Texture, TexCoord, ivec2( 0, -3));  vec4 c04 = textureOffset(Texture, TexCoord, ivec2( 1, -3));  vec4 c05 = textureOffset(Texture, TexCoord, ivec2( 2, -3));  vec4 c06 = textureOffset(Texture, TexCoord, ivec2( 3, -3));  vec4 c10 = textureOffset(Texture, TexCoord, ivec2(-3, -2));  vec4 c11 = textureOffset(Texture, TexCoord, ivec2(-2, -2));  vec4 c12 = textureOffset(Texture, TexCoord, ivec2(-1, -2));  vec4 c13 = textureOffset(Texture, TexCoord, ivec2( 0, -2));  vec4 c14 = textureOffset(Texture, TexCoord, ivec2( 1, -2));  vec4 c15 = textureOffset(Texture, TexCoord, ivec2( 2, -2));  vec4 c16 = textureOffset(Texture, TexCoord, ivec2( 3, -2));  vec4 c20 = textureOffset(Texture, TexCoord, ivec2(-3, -1));  vec4 c21 = textureOffset(Texture, TexCoord, ivec2(-2, -1));  vec4 c22 = textureOffset(Texture, TexCoord, ivec2(-1, -1));  vec4 c23 = textureOffset(Texture, TexCoord, ivec2( 0, -1));  vec4 c24 = textureOffset(Texture, TexCoord, ivec2( 1, -1));  vec4 c25 = textureOffset(Texture, TexCoord, ivec2( 2, -1));  vec4 c26 = textureOffset(Texture, TexCoord, ivec2( 3, -1));  vec4 c30 = textureOffset(Texture, TexCoord, ivec2(-3, 0));  vec4 c31 = textureOffset(Texture, TexCoord, ivec2(-2, 0));  vec4 c32 = textureOffset(Texture, TexCoord, ivec2(-1, 0));  vec4 c33 = textureOffset(Texture, TexCoord, ivec2( 0, 0));  vec4 c34 = textureOffset(Texture, TexCoord, ivec2( 1, 0));  vec4 c35 = textureOffset(Texture, TexCoord, ivec2( 2, 0));  vec4 c36 = textureOffset(Texture, TexCoord, ivec2( 3, 0));  vec4 c40 = textureOffset(Texture, TexCoord, ivec2(-3, 1));  vec4 c41 = textureOffset(Texture, TexCoord, ivec2(-2, 1));  vec4 c42 = textureOffset(Texture, TexCoord, ivec2(-1, 1));  vec4 c43 = textureOffset(Texture, TexCoord, ivec2( 0, 1));  vec4 c44 = textureOffset(Texture, TexCoord, ivec2( 1, 1));  vec4 c45 = textureOffset(Texture, TexCoord, ivec2( 2, 1));  vec4 c46 = textureOffset(Texture, TexCoord, ivec2( 3, 1));  vec4 c50 = textureOffset(Texture, TexCoord, ivec2(-3, 2));  vec4 c51 = textureOffset(Texture, TexCoord, ivec2(-2, 2));  vec4 c52 = textureOffset(Texture, TexCoord, ivec2(-1, 2));  vec4 c53 = textureOffset(Texture, TexCoord, ivec2( 0, 2));  vec4 c54 = textureOffset(Texture, TexCoord, ivec2( 1, 2));  vec4 c55 = textureOffset(Texture, TexCoord, ivec2( 2, 2));  vec4 c56 = textureOffset(Texture, TexCoord, ivec2( 3, 2));  vec4 c60 = textureOffset(Texture, TexCoord, ivec2(-3, 3));  vec4 c61 = textureOffset(Texture, TexCoord, ivec2(-2, 3));  vec4 c62 = textureOffset(Texture, TexCoord, ivec2(-1, 3));  vec4 c63 = textureOffset(Texture, TexCoord, ivec2( 0, 3));  vec4 c64 = textureOffset(Texture, TexCoord, ivec2( 1, 3));  vec4 c65 = textureOffset(Texture, TexCoord, ivec2( 2, 3));  vec4 c66 = textureOffset(Texture, TexCoord, ivec2( 3, 3));  texel = c00 \* coefs[ 0] + c01 \* coefs[ 1] + c02 \* coefs[ 2] + c03 \* coefs[ 3] + c04 \* coefs[ 4] + c05 \* coefs[ 5] + c06 \* coefs[ 6] +  c10 \* coefs[ 7] + c11 \* coefs[ 8] + c12 \* coefs[ 9] + c13 \* coefs[10] + c14 \* coefs[11] + c15 \* coefs[12] + c16 \* coefs[13] +  c20 \* coefs[14] + c21 \* coefs[15] + c22 \* coefs[16] + c23 \* coefs[17] + c24 \* coefs[18] + c25 \* coefs[19] + c26 \* coefs[20] +  c30 \* coefs[21] + c31 \* coefs[22] + c32 \* coefs[23] + c33 \* coefs[24] + c34 \* coefs[25] + c35 \* coefs[26] + c36 \* coefs[27] +  c40 \* coefs[28] + c41 \* coefs[29] + c42 \* coefs[30] + c43 \* coefs[31] + c44 \* coefs[32] + c45 \* coefs[33] + c46 \* coefs[34] +  c50 \* coefs[35] + c51 \* coefs[36] + c52 \* coefs[37] + c53 \* coefs[38] + c54 \* coefs[39] + c55 \* coefs[40] + c56 \* coefs[41] +  c60 \* coefs[42] + c61 \* coefs[43] + c62 \* coefs[44] + c63 \* coefs[45] + c64 \* coefs[46] + c65 \* coefs[47] + c66 \* coefs[48];  }  void LoG1(void)  {  float coefs[49] = float[](  -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0,  -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0,  -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0,  -1.0, -1.0, -1.0, 48.0, -1.0, -1.0, -1.0,  -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0,  -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0,  -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0  );  f\_7x7(coefs);  }  void LoG2(void)  {  float coefs[49] = float[](  0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 0.0,  0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 0.0,  0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 0.0,  -1.0, -1.0, -1.0, 12.0, -1.0, -1.0, -1.0,  0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 0.0,  0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 0.0,  0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 0.0  );  f\_7x7(coefs);  }  void main(void)  {  switch(Filter)  {  case 1U:  LoG1();  break;  case 2U:  LoG2();  break;  default:  texel = texture(Texture, TexCoord);  break;  }  Color = texel;  }  );  This code creates the vertex shader and fragment shader respectively.  **2、How to achieve the display of titles, menus, texts, etc?**  Call the CFrmFontGLES and CFrmUserInterfaceGLES classes provided by the SDK framework to display the title, menu and text.  // Create the font  m\_pFont = new CFrmFontGLES();  if( FALSE == m\_pFont->Create( "Samples/Fonts/Tuffy12.pak" ) )  {  FrmLogMessage("ERROR: create m\_pFont failed\n");  return FALSE;  }  // Load the packed resources  CFrmPackedResourceGLES resource;  if( FALSE == resource.LoadFromFile( "Samples/Textures/Logo.pak" ) )  {  return FALSE;  }  void main()  {  vec2 fragCoord = gl\_FragCoord.xy;  float solid = texture(Obstacles, InverseSize \* fragCoord).x;  if (solid > 0.0f)  {  FragColor = vec4(0.0f);  return;  }  vec2 u = texture(VelocityTexture, InverseSize \* fragCoord).xy;  vec2 coord = InverseSize \* (fragCoord - TimeStep \* u);  FragColor = Dissipation \* texture(SourceTexture, coord);  }  );  // Create the logo texture  m\_pLogoTexture = resource.GetTexture( "Logo" );  // Setup the user interface  if( FALSE == m\_UserInterface.Initialize( m\_pFont, g\_strWindowTitle ) )  {  return FALSE;  }  m\_UserInterface.AddOverlay( m\_pLogoTexture->m\_hTextureHandle, -5, -5, m\_pLogoTexture->m\_nWidth, m\_pLogoTexture->m\_nHeight );  m\_UserInterface.AddTextString( (char \*)"Press \200 for Help", 1.0f, -1.0f );  The above code creates the CFrmFontGLES and CFrmUserInterfaceGLES objects, respectively, where the CFrmUserInterfaceGLES object can call the AddOverlay and AddTextString methods to display icons and text.   1. **How to display the wave rate?**   Create a CFrmTimer class object and call the CFrmUserInterfaceGLES object in the render function to display the smoke rate.  // Update the timer  m\_Timer.MarkFrame();  // Render the user interface  m\_UserInterface.Render( m\_Timer.GetFrameRate() );   1. **How to compile and generate APK?**   Compile thesource code  cd jni/  ndk-build -B  cd ..  Update the resource file  ./InstallAssets.sh  android update project -p . -t android-24  Generate the APK file  ant debug  **5、How to use Snapdragon Profiler software to analysis CPU and GPU utilization ?**   1. Connect the phone to Snapdragon Profiler software. 2. Select the Realtime mode. 3. Select CPU Utilization and GPU Utilization in process list, and the real-time utilization results will be displayed in the middle window.  |  | | --- | | **/home/zx/Pictures/2020-09-04 13-56-16 的屏幕截图.png2020-09-04 13-56-16 的屏幕截图** |  1. **How to use Snapdragon Profiler software to do OpenGL analysis?** 2. Connect the phone to Snapdragon Profiler software. 3. Change layout setting to OpenGL. 4. Select the Snapshot Capture mode. 5. Take Snapshot.  |  | | --- | | **/home/zx/Pictures/2020-09-04 13-56-50 的屏幕截图.png2020-09-04 13-56-50 的屏幕截图** | | |
| **Usage Instructions** | 1. Download code from github according to the repository from　“   https://github.com/ThunderSoft-XA/Adreno-GPU-SDK-PROJECT.git”.  2、Install Adreno GPU SDK v5.0.  3、Install Android NDK r17b.  4、Install Apache Ant 1.9.14.  5、Install Snapdragon Profiler.  6、Compile APK source code and install it to android device.  7、Open the APK, and you will enjoy it.  8、If you care about some performance indicators, you can use Snapdragon Profiler to view it. | |
| **Contributor(s) Info**  *Feel free to include headshots!* | Name | Title  Company |
| Zhen  [sunzhen@thundersoft.com](mailto:yangrong0925@thundersoft.com) | Thundersoft |
| Rong  [yangrong0925@thundersoft.com](mailto:yangrong0925@thundersoft.com) | Thundersoft |
| Wu  [kouzw0723@thundersoft.com](mailto:Kouzw0723@thundersoft.com) | Thundersoft |
| Qin  yansq0930@thundersoft.com | Thundersoft |
| Xu  zhangxu@thundersoft.com | Thundersoft |

––– Continued on next page –––

Filters and Tags for QDN projects page

|  |  |  |
| --- | --- | --- |
| **Platform/Hardware** | CSR 101x/102x Bluetooth  DragonBoard 410c  mangOH Red/Yellow | MDM920x LTE for IoT  QCA-402x WiFi/BLE/Zigbee  ☐     Qualcomm Robotics RBx Dev Kit |
| **Software Tools** | 3D Audio Plugin for Unity  Adreno GPU SDK  Hexagon DSP SDK | Neural Processing SDK for AI  Snapdragon Profiler |
| **Operating System** | Android  Linux  ThreadX RTOS | Ubuntu Core  Windows 10 IoT Core |
| **Cloud Services/Platform** | Sierra Wireless AirVantage  Gizwits Cloud Platform  AT&T M2X  IBM Bluemix | IBM Watson IoT  Microsoft Azure IoT  Amazon AWS IoT |
| **Skill Level Required** | Advanced  Beginner  Intermediate |  |
| **Areas of Focus** | 3D Printing & Modeling  Alexa Voice Service  Artificial Intelligence  Bluetooth  Computer Vision  Digital Signage  Education  Embedded  Gaming | Healthcare  IoT  Robotics  Security  Sensors  Smart Cities  Smart Home  Toys |

*By submitting your content (“Submission”), you are granting Qualcomm a royalty-free, perpetual, non-exclusive, unrestricted, worldwide license to: (a) post, use, copy, sublicense, adapt, transmit, publicly perform or display any such Submission, (b) use, reproduce, modify, adapt, publish, translate, create derivative works from, distribute, perform, play, host, communicate, make available and publish your Submission without restriction and (c) sublicense to third parties the unrestricted right to exercise any of the foregoing rights granted with respect to the Submission. The foregoing grants shall include the right to exploit any ideas, concepts, intellectual property, or proprietary rights in such Submission, including but not limited to rights under copyright, trademark, servicemark or patent laws under any relevant jurisdiction without Qualcomm owing any monies to you whatsoever. You represent and warrant that you own all right, title and interest in and to the Submission, or you have been granted sufficient rights in and to the Submission allowing the foregoing use of such Submission.*