Qualcomm Developer Project C865DK-BilateralFilter

Project Submission

|  |  |  |
| --- | --- | --- |
| **Attribution:** |  | |
| **Email address** | [zhangzz6687@thundersoft.com](mailto:zhangzz6687@thundersoft.com)  [yuandk0305@thundersoft.com](mailto:yuandk0305@thundersoft.com)  zhanglei0706@thundersoft.com | |
| **Project Title**\* | **C865DK-BilateralFilter** | |
| **Images**  *Upload up to 5 images of your project*  *Please submit/send the original JPEG/PNG files for all images included in the document* | C865DK.png  C865DK.png [alt tag: “C865DK-BilateralFilter using the C865DK which is designed with Qualcomm® Snapdragon™ SM8250 processor as development board.”] **Type-C.png**   |  | | --- | | **typc** |   [alt tag: “**using the type-c line to develop on** C865DK **development board.** ”] | |
| **Description**\*  *High level description of the project* ***(75 words or less)*** | Develop a demo application using the Adreno GPU SDK to display Bilateral Filter models on C865DK platform. | |
| **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 Bilateral Filter 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 |
| C865DK | https://www.thundercomm.com/app\_en/product/1598010487118378 |
| Type-c line | https://detail.tmall.com/item.htm?id=44425281296&ali\_refid=a3\_430582\_1006:1103572855:N:8BFxSxK119dzkfQCc2yGI2us815vvcUHETWnj5g1swo=:6399b40850a40201c56536531a885bcf&ali\_trackid=1\_6399b40850a40201c56536531a885bcf&spm=a230r.1.14.11 |
| 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 |
| Android SDK | dl.google.com/android/android-sdk\_r24.4.1-linux.tgz |
| Snapdragon Profiler | https://github.com/tzutalin/dlib-android |
|  |  |
| **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](https://github.com/ThunderSoft-XA/demo-Smart-Motion-detector)/C865DK- BilateralFilter |
|  |  |
|  |  |
|  |  |
| **Additional Resources**  *List related links or resources such as websites, videos, presentations, or other materials* | Resource Title | Link or File Name (and provide file) |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Build / Assembly Instructions** | Parts used  1. Android device with C865DK installed the APK compiled by Android NDK, used to run APK and see image filter display effect. 2. PC with Ubuntu 16.04. 3. Type-C data cable. 4. Adreno GPU SDK v5.0. 5. Android SDK 24.4.1 6. Android NDK r17b. 7. Apache Ant 1.9.14. 8. 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 and install it to PC.  3.Download Android NDK r17b and install it to PC.  4.Download Apache Ant 1.9.14 and install it to PC.  5.Download Snapdragon Profiler from <https://developer.qualcomm.com/software/snapdragon-profiler>, and install it to PC.  6.Realize the Bilateral Filter shader.  7.Call the Bilateral Filter shader in the APK.  8.Display FPS in the APK.  9.Realize required functions step by step.  10.Choose android platform and connect android device, then complies the project, then install the APK.  11.Open it and you’ll enjoy.  12.if you care about some performance indicators, you can use Snapdragon Profiler to view it. | |
|  | Sample outline:  How does it work?  **1.How to realize the shader？**  In order to enable OpenGL to achieve Bilateral Filter, 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;  }  );  //}} Vertex Shaders  **Fragment Shader：**  //{{ Fragment Shaders  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 = STRINGIZE(  uniform sampler2D Texture;  in vec4 FragColor;  in vec2 TexCoord;  layout (location = 0) out vec4 Color;  void main(void)  {  vec4 texel = texture(Texture, TexCoord);  Color = texel;  }  );  const GLchar \* fs\_frag = STRINGIZE(  uniform sampler2D Texture;  in vec4 FragColor;  in vec2 TexCoord;  layout (location = 0) out vec4 Color;  void main(void)  {  Color = FragColor;  }  );  const GLchar \* fs\_tex\_filter = STRINGIZE(  float normpdf(in float x, in float sigma)  {  return 0.39894\*exp(-0.5\*x\*x/(sigma\*sigma))/sigma;  }  float normpdf3(in vec3 v, in float sigma)  {  return 0.39894\*exp(-0.5\*dot(v,v)/(sigma\*sigma))/sigma;  }  uniform sampler2D Texture;  uniform uint Filter;  in vec4 FragColor;  in vec2 TexCoord;  layout (location = 0) out vec4 Color;  void main(void) {  vec2 iResolution = vec2(372, 558);  vec3 c = texture(Texture, TexCoord.xy).rgb;  int kSize = 7;  float kernel[15];  vec3 bfinal\_colour = vec3(0.0);  float bZ = 0.0;  //create the 1-D kernel  for (int j = 0; j <= kSize; ++j) {  kernel[kSize+j] = kernel[kSize-j] = normpdf(float(j), 10.0);  }  vec3 cc;  float gfactor;  float bfactor;  float bZnorm = 1.0/normpdf(0.0, 0.1);  //read out the texels  for (int i=-kSize; i <= kSize; ++i)  {  for (int j=-kSize; j <= kSize; ++j)  {  cc = texture(Texture, TexCoord.xy+vec2(float(i),float(j)) / iResolution.xy).rgb;  // compute both the gaussian smoothed and bilateral  gfactor = kernel[kSize+j]\*kernel[kSize+i];  bfactor = normpdf3(cc-c, 0.1)\*bZnorm\*gfactor;  bZ += bfactor;  bfinal\_colour += bfactor\*cc;  }  }  vec4 bc = vec4(bfinal\_colour/bZ, 1.0);    switch(Filter)  {  case 1U:  break;  default:  bc = texture(Texture, TexCoord);  break;  }  Color = bc;  }  );  **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 fps?**   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() );  **4. How to compile and generate APK?**  Compile the source 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 | |
| **Usage Instructions** | Sample outline:   1. Install APK to C865DK device   adb install BilateralFilter.apk   1. Start app.   1) start “BilateralFilter” app  ic_launcher.png  2)Check the screen before Bilateral Filter  2021-04-14 09-56-08屏幕截图.png  3)Check the screen after Bilateral Filter  2021-04-14 09-56-27屏幕截图.png  3.Use Snapdragon Profiler tools  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.  2021-04-13 20-47-34屏幕截图.png  4)Change layout setting to OpenGL.  5)Select the Snapshot Capture mode.  6)Take Snapshot.  2021-04-13 20-46-42屏幕截图.png | |
| **Contributor(s) Info**  *Feel free to include headshots!* | Name | Title  Company |
|  |  |
|  |  |
|  |  |

––– 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  √     Turbox™ C865 Development 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.*