  Android手机有个挺好的功能，它允许你往桌面上放窗口小部件（widget），有一个叫相框的小部件，可以让你选择一张相片，截取一部分，放在相框里。我桌面上就放了几个相框，里面是我女儿的照片，隔阵子换一换，挺喜欢。这次的实例受相框小部件启发而成，我称之为挖头像，先看看运行效果。

**运行效果**

    电脑上的运行效果如图1：

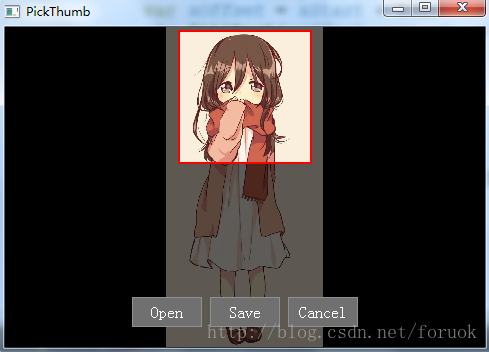


                           图1 电脑挖头像效果图

    Android手机上运行效果如图2：



**项目创建**

    项目创建过程参考《[Qt Quick 之 Hello World 图文详解](http://blog.csdn.net/foruok/article/details/28850879)》，安卓配置参考《[Windows下Qt 5.2 for Android开发入门](http://blog.csdn.net/foruok/article/details/17305697)》和《[Qt on Android：图文详解Hello World全过程](http://blog.csdn.net/foruok/article/details/23528293)》。

    项目名称是PickThumb，Android包名是an.qt.PickThumb，其它的木啥咧。

**源码分析**

**C++代码**

    为了能够让PickThumb正常退出，我给QGuiApplication安装了事件过滤器，过滤BACK按键。下面是main.cpp文件：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/38778233)[在CODE上查看代码片](https://code.csdn.net/snippets/456841)

1. #include <QGuiApplication>
2. #include <QQmlApplicationEngine>
3. #include <QKeyEvent>
5. class KeyBackQuit: public QObject
6. {
7. public:
8. KeyBackQuit(QObject \*parent = 0)
9. : QObject(parent)
10. {}
12. **bool** eventFilter(QObject \*watched, QEvent \* e)
13. {
14. switch(e->type())
15. {
16. case QEvent::KeyPress:
17. if( ((QKeyEvent\*)e)->key() == Qt::Key\_Back )
18. {
19. e->accept();
20. return true;
21. }
22. break;
23. case QEvent::KeyRelease:
24. if( ((QKeyEvent\*)e)->key() == Qt::Key\_Back )
25. {
26. e->accept();
27. qApp->quit();
28. return true;
29. }
30. break;
31. default:
32. break;
33. }
34. return QObject::eventFilter(watched, e);
35. }
37. };
39. **int** main(**int** argc, **char** \*argv[])
40. {
41. QGuiApplication app(argc, argv);
42. app.installEventFilter(new KeyBackQuit);
44. QQmlApplicationEngine engine;
45. engine.load(QUrl(QStringLiteral("qrc:///main.qml")));
47. return app.exec();
48. }

#include <QGuiApplication>

#include <QQmlApplicationEngine>

#include <QKeyEvent>

class KeyBackQuit: public QObject

{

public:

KeyBackQuit(QObject \*parent = 0)

: QObject(parent)

{}

bool eventFilter(QObject \*watched, QEvent \* e)

{

switch(e->type())

{

case QEvent::KeyPress:

if( ((QKeyEvent\*)e)->key() == Qt::Key\_Back )

{

e->accept();

return true;

}

break;

case QEvent::KeyRelease:

if( ((QKeyEvent\*)e)->key() == Qt::Key\_Back )

{

e->accept();

qApp->quit();

return true;

}

break;

default:

break;

}

return QObject::eventFilter(watched, e);

}

};

int main(int argc, char \*argv[])

{

QGuiApplication app(argc, argv);

app.installEventFilter(new KeyBackQuit);

QQmlApplicationEngine engine;

engine.load(QUrl(QStringLiteral("qrc:///main.qml")));

return app.exec();

}

    KeyBackQuit类重写eventFilter()方法来过滤Key\_Back按键，调用QCoreApplication的quit()方法退出应用。过滤器在main()函数中被安装到QGuiApplication实例上。

**QML代码分析**

    该主角登场了，main.qml文件有200多行代码，内容如下：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/38778233)[在CODE上查看代码片](https://code.csdn.net/snippets/456841)

1. import QtQuick 2.2
2. import QtQuick.Window 2.1
3. import QtQuick.Controls 1.2
4. import QtQuick.Controls.Styles 1.2
5. import QtQuick.Dialogs 1.1
7. Window {
8. visible: true
9. width: 480;
10. height: 320;
11. minimumHeight: 320;
12. minimumWidth: 480;
13. color: "black";
15. onWidthChanged: mask.recalc();
16. onHeightChanged: mask.recalc();
18. Image {
19. id: source;
20. anchors.fill: parent;
21. fillMode: Image.PreserveAspectFit;
22. visible: false;
23. asynchronous: true;
24. onStatusChanged: {
25. if(status == Image.Ready){
26. console.log("image loaded");
27. mask.recalc();
28. }
29. }
30. }
32. FileDialog {
33. id: fileDialog;
34. title: "Please choose an Image File";
35. nameFilters: ["Image Files (\*.jpg \*.png \*.gif)"];
36. onAccepted: {
37. source.source = fileDialog.fileUrl;
38. }
39. }

42. Canvas {
43. id: forSaveCanvas;
44. width: 128;
45. height: 128;
46. contextType: "2d";
47. visible: false;
48. z: 2;
49. anchors.top: parent.top;
50. anchors.right: parent.right;
51. anchors.margins: 4;
53. property var imageData: null;
54. onPaint: {
55. if(imageData != null){
56. context.drawImage(imageData, 0, 0);
57. }
58. }
60. function setImageData(data){
61. imageData = data;
62. requestPaint();
63. }
64. }
66. Canvas {
67. id: mask;
68. anchors.fill: parent;
69. z: 1;
70. property real w: width;
71. property real h: height;
72. property real dx: 0;
73. property real dy: 0;
74. property real dw: 0;
75. property real dh: 0;
76. property real frameX: 66;
77. property real frameY: 66;
79. function calc(){
80. var sw = source.sourceSize.width;
81. var sh = source.sourceSize.height;
82. if(sw > 0 && sh > 0){
83. if(sw <= w && sh <=h){
84. dw = sw;
85. dh = sh;
86. }else{
87. var sRatio = sw / sh;
88. dw = sRatio \* h;
89. if(dw > w){
90. dh = w / sRatio;
91. dw = w;
92. }else{
93. dh = h;
94. }
95. }
96. dx = (w - dw)/2;
97. dy = (h - dh)/2;
98. }
99. }
101. function recalc(){
102. calc();
103. requestPaint();
104. }
106. function getImageData(){
107. return context.getImageData(frameX - 64, frameY - 64,
108. 128, 128);
109. }
111. onPaint: {
112. var ctx = getContext("2d");
113. if(dw < 1 || dh < 1) {
114. ctx.fillStyle = "#0000a0";
115. ctx.font = "20pt sans-serif";
116. ctx.textAlign = "center";
117. ctx.fillText("Please Choose An Image File",
118. width/2, height/2);
119. return;
120. }
121. ctx.clearRect(0, 0, width, height);
122. ctx.drawImage(source, dx, dy, dw, dh);
123. var xStart = frameX - 66;
124. var yStart = frameY - 66;
125. ctx.save();
126. ctx.fillStyle = "#a0000000";
127. ctx.fillRect(0, 0, w, yStart);
128. var yOffset = yStart + 132;
129. ctx.fillRect(0, yOffset, w, h - yOffset);
130. ctx.fillRect(0, yStart, xStart, 132);
131. var xOffset = xStart + 132;
132. ctx.fillRect(xOffset, yStart, w - xOffset, 132);
134. //see through area
135. ctx.strokeStyle = "red";
136. ctx.fillStyle = "#00000000";
137. ctx.lineWidth = 2;
138. ctx.beginPath();
139. ctx.rect(xStart, yStart, 132, 132);
140. ctx.fill();
141. ctx.stroke();
142. ctx.closePath ();
143. ctx.restore();
144. }
145. }
147. MultiPointTouchArea {
148. anchors.fill: parent;
149. minimumTouchPoints: 1;
150. maximumTouchPoints: 1;
151. touchPoints:[
152. TouchPoint{
153. id: point1;
154. }
155. ]
157. onUpdated: {
158. mask.frameX = point1.x;
159. mask.frameY = point1.y;
160. mask.requestPaint();
161. }
162. onReleased: {
163. forSaveCanvas.setImageData(mask.getImageData());
164. actionPanel.visible = true;
165. }
166. onPressed: {
167. actionPanel.visible = false;
168. }
169. }
171. Component {
172. id: flatButton;
173. ButtonStyle {
174. background: Rectangle{
175. implicitWidth: 70;
176. implicitHeight: 30;
177. border.width: control.hovered ? 2: 1;
178. border.color: control.hovered ? "#c0c0c0" : "#909090";
179. color: control.pressed ? "#a0a0a0" : "#707070";
180. }
181. label: Text {
182. anchors.fill: parent;
183. font.pointSize: 12;
184. horizontalAlignment: Text.AlignHCenter;
185. verticalAlignment: Text.AlignVCenter;
186. text: control.text;
187. color: (control.hovered && !control.pressed) ?
188. "blue": "white";
189. }
190. }
191. }
193. Row {
194. anchors.horizontalCenter: parent.horizontalCenter;
195. anchors.bottom: parent.bottom;
196. anchors.bottomMargin: 20;
197. id: actionPanel;
198. z: 5;
199. spacing: 8;
200. Button {
201. style: flatButton;
202. text: "Open";
203. onClicked: fileDialog.open();
204. }
205. Button {
206. style: flatButton;
207. text: "Save";
208. onClicked: {
209. forSaveCanvas.save("selected.png");
210. actionPanel.visible = false;
211. }
212. }
213. Button {
214. style: flatButton;
215. text: "Cancel";
216. onClicked: actionPanel.visible = false;
217. }
218. }
219. }

import QtQuick 2.2

import QtQuick.Window 2.1

import QtQuick.Controls 1.2

import QtQuick.Controls.Styles 1.2

import QtQuick.Dialogs 1.1

Window {

visible: true

width: 480;

height: 320;

minimumHeight: 320;

minimumWidth: 480;

color: "black";

onWidthChanged: mask.recalc();

onHeightChanged: mask.recalc();

Image {

id: source;

anchors.fill: parent;

fillMode: Image.PreserveAspectFit;

visible: false;

asynchronous: true;

onStatusChanged: {

if(status == Image.Ready){

console.log("image loaded");

mask.recalc();

}

}

}

FileDialog {

id: fileDialog;

title: "Please choose an Image File";

nameFilters: ["Image Files (\*.jpg \*.png \*.gif)"];

onAccepted: {

source.source = fileDialog.fileUrl;

}

}

Canvas {

id: forSaveCanvas;

width: 128;

height: 128;

contextType: "2d";

visible: false;

z: 2;

anchors.top: parent.top;

anchors.right: parent.right;

anchors.margins: 4;

property var imageData: null;

onPaint: {

if(imageData != null){

context.drawImage(imageData, 0, 0);

}

}

function setImageData(data){

imageData = data;

requestPaint();

}

}

Canvas {

id: mask;

anchors.fill: parent;

z: 1;

property real w: width;

property real h: height;

property real dx: 0;

property real dy: 0;

property real dw: 0;

property real dh: 0;

property real frameX: 66;

property real frameY: 66;

function calc(){

var sw = source.sourceSize.width;

var sh = source.sourceSize.height;

if(sw > 0 && sh > 0){

if(sw <= w && sh <=h){

dw = sw;

dh = sh;

}else{

var sRatio = sw / sh;

dw = sRatio \* h;

if(dw > w){

dh = w / sRatio;

dw = w;

}else{

dh = h;

}

}

dx = (w - dw)/2;

dy = (h - dh)/2;

}

}

function recalc(){

calc();

requestPaint();

}

function getImageData(){

return context.getImageData(frameX - 64, frameY - 64,

128, 128);

}

onPaint: {

var ctx = getContext("2d");

if(dw < 1 || dh < 1) {

ctx.fillStyle = "#0000a0";

ctx.font = "20pt sans-serif";

ctx.textAlign = "center";

ctx.fillText("Please Choose An Image File",

width/2, height/2);

return;

}

ctx.clearRect(0, 0, width, height);

ctx.drawImage(source, dx, dy, dw, dh);

var xStart = frameX - 66;

var yStart = frameY - 66;

ctx.save();

ctx.fillStyle = "#a0000000";

ctx.fillRect(0, 0, w, yStart);

var yOffset = yStart + 132;

ctx.fillRect(0, yOffset, w, h - yOffset);

ctx.fillRect(0, yStart, xStart, 132);

var xOffset = xStart + 132;

ctx.fillRect(xOffset, yStart, w - xOffset, 132);

//see through area

ctx.strokeStyle = "red";

ctx.fillStyle = "#00000000";

ctx.lineWidth = 2;

ctx.beginPath();

ctx.rect(xStart, yStart, 132, 132);

ctx.fill();

ctx.stroke();

ctx.closePath ();

ctx.restore();

}

}

MultiPointTouchArea {

anchors.fill: parent;

minimumTouchPoints: 1;

maximumTouchPoints: 1;

touchPoints:[

TouchPoint{

id: point1;

}

]

onUpdated: {

mask.frameX = point1.x;

mask.frameY = point1.y;

mask.requestPaint();

}

onReleased: {

forSaveCanvas.setImageData(mask.getImageData());

actionPanel.visible = true;

}

onPressed: {

actionPanel.visible = false;

}

}

Component {

id: flatButton;

ButtonStyle {

background: Rectangle{

implicitWidth: 70;

implicitHeight: 30;

border.width: control.hovered ? 2: 1;

border.color: control.hovered ? "#c0c0c0" : "#909090";

color: control.pressed ? "#a0a0a0" : "#707070";

}

label: Text {

anchors.fill: parent;

font.pointSize: 12;

horizontalAlignment: Text.AlignHCenter;

verticalAlignment: Text.AlignVCenter;

text: control.text;

color: (control.hovered && !control.pressed) ?

"blue": "white";

}

}

}

Row {

anchors.horizontalCenter: parent.horizontalCenter;

anchors.bottom: parent.bottom;

anchors.bottomMargin: 20;

id: actionPanel;

z: 5;

spacing: 8;

Button {

style: flatButton;

text: "Open";

onClicked: fileDialog.open();

}

Button {

style: flatButton;

text: "Save";

onClicked: {

forSaveCanvas.save("selected.png");

actionPanel.visible = false;

}

}

Button {

style: flatButton;

text: "Cancel";

onClicked: actionPanel.visible = false;

}

}

}

    代码的逻辑是这样的：点击“Open”按钮打开一个对话框，用户选择一张图片，使用隐藏的Image对象加载，加载成功后触发Canvas对象绘制图片；当用户用手指（或按下鼠标左键）拖动时，选中框中心跟随手指移动，框内图像是正常亮度；当用户抬起手指后，弹出操作菜单，如选择“Save”，则通过一个隐藏的Canvas把选中区域的图像保存到文件中。

    QML中用到的Row、Button、ButtonStyle、Component、Image、FileDialog等我都有文章讲过，参考我的专栏《[Qt Quick简明教程](http://blog.csdn.net/column/details/qtquick.html)》；MultiPointTouchArea和Canvas没讲过，参考Qt帮助吧。这里咱单说“整个照片变暗而唯有选中框内正常显示”这种效果的实现。  
    我定义了一个id为mask的Canvas，它使用id为source的Image对象绘制图片。图片在最底层绘制，然后在它上面绘制使用透明色填充的矩形，于是图片就变暗了。整个Canvas被分成一个“回”字形，中间是完全透明的矩形，周围是半透明的。半透明部门由顶部、底部、左面、右面四个矩形组成，分别填充即可。  
    图片是按比例显示的，等图片加载成功后，先计算了绘制时需要的目标矩形，绘制时直接引用，避免重复计算。而桌面版本为了适应窗口大小变化，实现了onWidthChanged和onHeightChanged两个信号处理器来更新绘制参数。  
    当用户选择保存时，把mask的透明区域内的像素挖出来（getImageData），生成一个CanvasImageData对象，交给另一个Canvas对象去显示，调用它的save()方法把内容写入到文件。

    这就是全部了。