EC535 Introduction to Embedded Systems
Project Report, Fall 2014

Tetris on Touchscreen

Player 1

Shulin Qian, chrisq@bu.edu
Yuchen Huang, hyc@bu.edu
Xin Guo, xinguo@bu.edu

Abstract

Tetris is a well-known and classic video game originally designed and programmed by Alexey Pajitnov. In this project, we implemented this game on a touchscreen which was driven by Gumstix. In addition, users can also control the game by the virtual keyboard on their smartphones which are connected to the system by Bluetooth.

1. Introduction

Tetris is the first entertainment software to be exported from the USSR to the US and published by Spectrum HoloByte for Commodore 64 and IBM PC. This game is a popular use of tetrominoes, the four-element special case of polyominoes. As shown in Figure 1.

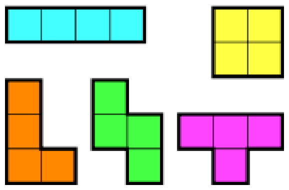

Figure 1 Tetrominoes

To pay our tributes to this game, we decided to implement the touchscreen version of it. And, since this is an era of smartphones, we choose them to be the virtual controllers to play this game through Bluetooth connection.

Our project consists of two components. The main component is the LCD screen. Except for the main game area, UI and all the operational buttons are also put on the screen which is shown in following.

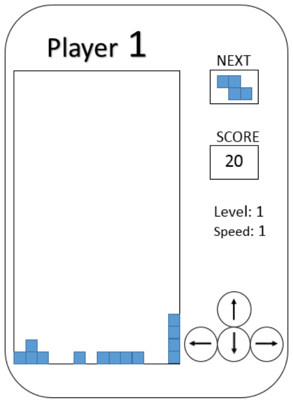

Figure 2 Layout Design

On this screen, users can adjust the falling speed of tetrominoes, change the background color and, of course, play this classic game.

The other component is the virtual keyboard which contains four arrow buttons. The functions of them are:

- Up: rotate the tetromino which is falling by 90 degrees
- Down: drop the tetromino
- Left: left-shift the tetromino
- Right: right-shift the tetromino

We accomplished to display the UI on the touch screen, connect the virtual keyboard which was on the smartphone to this system. Players can choose either of these two ways to enjoy this game! We managed to improve the stability of the Bluetooth connection.

2. Design Flow

Our project includes three components. LCD display, Android virtual keyboard development, and Bluetooth communication which is shown below.

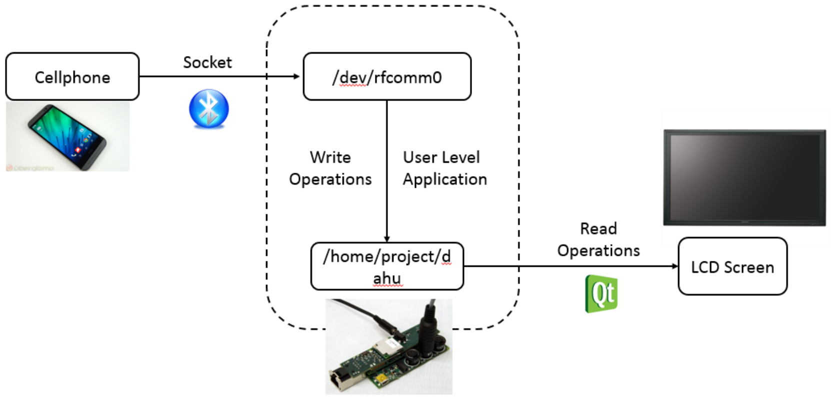

Figure 3 Design Flow

- The entire game was implemented on the LCD screen, which was developed by Qt (C++).

- The virtual keboard was implemented as an Android application which was developed by Java.
- The smartphone sent the player's operations to /dev/rfcomm0 through socket connection, and the read/write module wrote those command to /home/project/dahu in the form of strings.

Components impementation is shown in Table 1

Table 1 Division of Labor

Shulin Qian	Button functionalities: UP, DOWN, LEFT, RIGHT, Pause/Continue, STOP, Difficulty adjustment, Edge detection
Yuchen Huang	Game area display, Background change, Full rows removing & score calculation
Xin Guo	Bluetooth socket connection, read/write module

3. Project Details

a. Bluetooth Socket connection

Our project includes the Bluetooth remote controller which is based on android cellphone. We decided use the android java language to compile that function via Eclipse.

Firstly, the program will set up a Bluetooth adapter and make a UUID for our Gumstix board and get the device information from the Bluetooth adapter.

Secondly, building a socket between the Gumstix Bluetooth and cellphone Bluetooth which is the channel we transmit the information.

Thirdly, we catch the signal if any buttons were pressed in order to know the users' command and compile those commands into different numbers.

Finally, program may transmit the different contents to give the information to the Gumstix board and waiting for the new commands from the users. Data flow is shown in Figure 4.

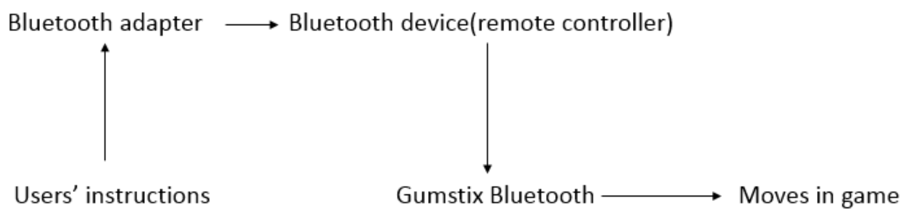

Figure 4 Dataflow

b. Read/Write Module

Read/Write Module is using to deal with the information from the remote controller and transmit the commands information to the QT part which is the game part. Actually this part copy the information and only put the useful information into the /home/root/project/dahu.

Firstly, program should check if Gumstix have connected to the remote controller which means Gumstix have the file (/dev/rfcomm0).

Secondly, if program get the true return value then it should read the information from the specified file using command fread(). After that the information should be copy to the file /home/root/project/dahu.

Finally, closing all sockets and make sure the safety of the information and the integrity of all the files to ensure all information would be correct.

c. Touchscreen Module

This module was implemented by Qt entirely. Following functionalities are included:

Game area display

This component was initialed by function GameArea::init_gameArea() which included several basic settings such as the position, the color and the size.

We created a timer whose callback function is Widget::timer_upDate() which moved the tetromino to the next step, that is, changed the coordinate of the tetromino and refreshed the entire screen (GameArea::draw_gameArea). If no buttons were pressed before the timer expired, the tetromino dropped one block. As shown in Figure 5.

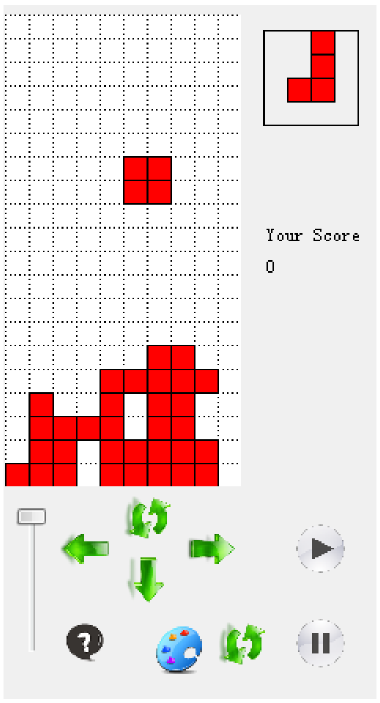

Figure 5 Main Screen

Game functional buttons

Four basic game functional buttons were displayed on the screen which provided the basic operations for the player.

- Up: Rotate the current tetromino by 90 degrees (GameArea::do_itemChange()) which was implemented by Widget::on_pushButton_12_clicked()

- Down: Drop the current tetromino to the bottom of the game area which was implemented by Widget::on_pushButton_8_clicked()

- Left: Left-shift the current tetromino which was implemented by Widget::on_pushButton_11_clicked()

- Right: Right-shift the current tetromino which was implemented by Widget::on_pushButton_10_clicked()

Game control buttons

Four game control buttons were displayed on the screen which provided the game process control for the player.

- Pause/Continue: Pause/continue the game which was implemented by stop the countdown of the timer we mentioned above. The function is Widget::on_pushButton_3_clicked()

- Stop/Start: Abandon current game and restart a new game from beginning. This was implemented by initialize the game. The function is Widget::on_pushButton_2_clicked()

- Background color change: Change the UI theme which was implemented by reset the color of the game area. The function is Widget::on_pushButton_4_clicked(). In this function setGetAreaColor(), setBoxBrushColor(), setBoxPenColor were used to change the color of the background, the little boxes, and the edges of boxes.

- Speed adjustment: This was implemented by a horizontal slide bar which had five different settings. It change the falling speed of the tetromino by set the period of the timer above.

Full rows removing, score calculation and edge detection

- Full rows removing: Check the game area block by block every time when the timer expired. Record the row number of the full rows. Remove them when the game area refreshed. This was implemented by function GameArea::clearRow()

- Score calculation: If there were rows cleared, added the corresponding score into the global variable

- Edge detection: Check the current tetromino was on the left/right edge of the game area, if it was, disable the respond of left/right button click. This was judged by function GameArea::isMoveLeft()/GameArea::isMoveRight()

- End detection: Check if the current tetromino could still fall. This was implemented by check if all the blocks under current tetromino were occupied, if yes, lock the coordinate and refreshed the entire game area when the timer expired. The function is GameArea::isMovedEnd(). As shown in Figure 6.

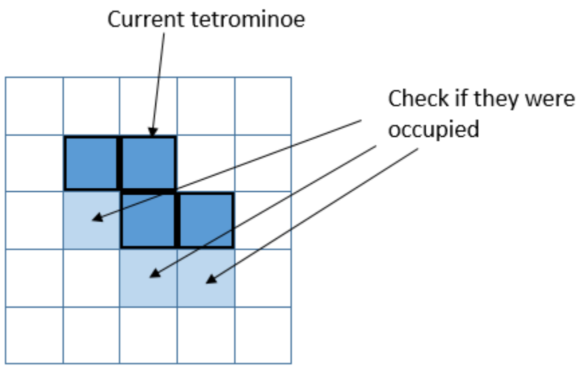

Figure 6 Bottom Detection

- Top detection: Check if there was enough space to put current tetromino. This was implemented by check if the coordinate of the top block of current tetromino was out of the upper bound of game area, if yes, set the global over tag to "true" and ended game and printed the "Game Over" warning.

Instruction reception

We created another timer, timer2, whose callback function was to check if any button was pressed and the instruction in file /home/root/project/dahu, executed the instruction, and refreshed the game area when the timer expired. The period of this timer is 10 milliseconds, namely, we scanned the instruction input every 10 milliseconds. The flow chart is shown in Figure 7. To simplify the chart, we do not take "Stop", "Change Background", and "Pause/Continue" into consideration.

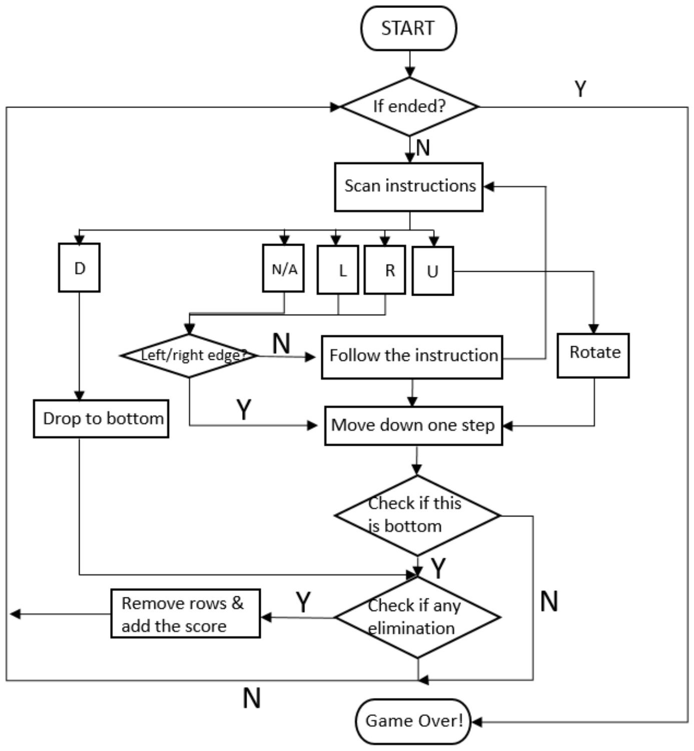

Figure 7 System Logic

Challenge and solution

The challenge in the instruction transmission was to distinguish between the case the player pressed the same button several times and the case the player press this button once and did nothing for next several periods. Our solution was to set a unique ID for each instruction and recorded the ID for the current instruction. After scanning the instruction's content and instruction's ID, we discarded this instruction and just moved the tetrominoe down one block and did nothing else if the ID was the identical with the previous one.

4. Summary

Our project implemented a touchscreen version of Tetris. The basic functionalities of our version include left/right shift, rotation, and drop of the tetrominoes and "pause/continue", "stop". Extra functionalities include background color switch and difficulty adjustment.

In addition, players can also use their smartphones to control the tetromino which are connected by the Bluetooth.

Overall, all the functionalities in the proposal have been implemented but the stability of the Bluetooth connection still needs to be improved.

References

[1] Qt Designer Manual. http://qt-project.org/doc/qt-4.8/designer-manual.html
[2] Gumstix Users. http://wiki.gumstix.org/index.php?title=Main_Page
[3] Android API Classes. http://developer.android.com/reference/packages.html
[4] Java API. http://docs.oracle.com/javase/7/docs/api/