CS 296 PROJECT : QUBIC

by

Arjun R. (05005025)
Amod Jog (05005009)
Rahul Bajaj (05005006)
Saurabh Chakradeo (05010024)
B S Raj Meena (05D05022)

Project Guide : Prof. S.Sudarshan
Acknowledgements:

Firstly, we would like to thank the technology of the internet which helped us get useful references easily and quickly. 'Long live the Internet.'

We acknowledge the help we obtained from Nokia Developer Forum and Sun Java Developer Forum for the initial pointers in writing J2ME softwares.

We also thank our seniors for providing useful hints and advice regarding bluetooth code development.
Table Of Contents

1. Introduction.

2. Overall Project Structure.

3. Implementation Details

4. Bluetooth implementation.

5. User Interface and Graphics.

6. Game Logic and AI

7. How To Play

8. Conclusion.
Introduction

Qubic is the brand name of a four-in-a-row game played in a 444 matrix by Parker Brothers in the late 1960s. Players alternately place pieces to get four in a row. There are 76 winning lines. It was weakly solved by Oren Patashnik (1980) and then solved again by Victor Allis using proof-number search. The Game was in a Microsoft Windows Game Pack in the 1990s.

We have developed a multiplayer cellphone version of this game which includes 3D graphics. The multiplayer part achieves communication between two mobile phones through the bluetooth technology and is implemented using the JSR82 API. The game scores on providing good graphical features enhancing the in-game experience by providing both 3D and 2D user interfaces. We have included the classical 4 layered 2D interface along with a dynamic mutable interfaces which helps the user visualise the cube better. These interfaces were implemented using the javax.microedition.m3g package.

This game also supports single player vs computer game. This has been implemented using a fairly simple and naive AI. The multiplayer feature is enabled through bluetooth. In the multiplayer mode the user has an option of either hosting a game or to join an existing server.

This project has been done keeping in mind all the standard 'Best Practises' making it a very well designed easy to extend application. A few of its salient features are

1. The code has been written in a complete modular way and every major part of the project is independent of the others.

2. The code is written in such a way that it is completely reusable. For example, the bluetooth module can be used in any other application for text communication between mobile devices.

3. There are many functions which are independant of the underlying algorithms, so that, the implementation style can be changed without affecting the structure of the program. For example, the game engine uses an algorithm to compute the computer’s move which can be totally replaced without changing any other part.
Implementation Details

1 Bluetooth Implementation

The Bluetooth module contains two major classes, BluetoothServer and BluetoothClient. The module also contains two interfaces BluetoothListener and BluetoothClientListener which are used to communicate with the BluetoothServer and the BluetoothClient respectively. The BluetoothServer has a method called btinit() which is used to initialize bluetooth services on the phone. BluetoothServer which extends Thread waits for a client to connect when started. Once a client connects, the server opens two separate threads for input and output. These are implemented by the BluetoothServerInput and BluetoothServerOutput classes which are nested inside the BluetoothServer class.

The BluetoothServer class also contains a member called listener of the type BluetoothListener. When a message is received by the BluetoothServerInput, it is passed on to the listener. The server also has a send(String s) method which is used to send strings to the client. It has a destroy() method which closes all connections to the client and cleans up.

BluetoothClientListener is interface used to communicate with BluetoothClient. It has method for addDevice(), searchCompleted() and messageReceived(). When a new device is found, the addDevice() is called. When all the devices have been listed then searchCompleted() is called. The messageReceived() method handles the messages received by the client from the server.

The BluetoothClient class extends a Thread. When started it first searches and discovers the active bluetooth devices running Qubic in the vicinity and informs the BluetoothClientListener about the same. It also contains two input and output thread classes which are BluetoothClientInput and BluetoothClientOutput which handle the communication with the selected server.
2 User Interface The user interface is mainly composed by the Disp class. When the game is started it gives the user three options.

a. Create a new game: If the user decides to create a new game, the user interface starts a new BluetoothServer and waits for connections from other clients.

b. Join an existing game: When the user chooses to join an existing game, the user interface makes a new BluetoothClient and instructs it to search for servers which are running Qubic. When it receives a list of prospective servers it presents it to the user on a menu and the user can choose one. Then the game begins.

c. Play against AI: The user interface directly starts a game with computer as the second player.

The Disp class now acts as a mediator between the user and the server/client and the game logic. For this, it uses two nested classes called GameBluetoothListener and GameBluetoothClientListener. These get the message from the server/client and call the right function later. To communicate with the user, the Disp class contains a CallBackCanvas which calls a keyAction() function with the keycode. It is then decided what action to take.

3 Game Graphics The Disp class also contains implementation of three dimensional and two dimensional graphics. The CallBackCanvas calls the paint() function of Disp class every time it is repainted. In this paint() function the appropriate graphics is rendered onto the screen.

a. Three Dimensional Graphics A m3g graphics World is created and the appropriate objects are added to it to render the 3D qubic board structure. On request from the user the structure can be rotate and viewed in different angles. A move can be made by pressing the appropriate key.

b. Two Dimensional Graphics 2D graphics are created using the drawString, drawRect and drawLine methods of the Graphics class.
4 **Game Logic and AI:** There are 76 distinct ways of winning the Qubic game. All decisions pertaining to logic is taken in our GameCube class. We maintain a 4x4x4 global array 'a' (global to the logic checking class) where every move of the players is stored. In the GameCube class we have a method `CheckWin` that checks whether any one of the players have completed one of the 76 possibilities, thereby winning the game. The method `GameMove` move registers the move made by the user and puts the user id in the a[i][j][k] where i,j,k is the selected coordinate by the player.

For the AI part we have a three step procedure (its a naive algorithm but it can be improved with ease) implemented using `CompMove` method. It works as follows,

*Firstly*, we check if there exists a winning move for the computer. If so, that move is executed.

*Secondly*, we if the computer doesn’t have a winning move then he sees whether the human has a winning move. If so, he blocks that move.

*Thirdly*, if both the first and second cases are not satisfied then it randomly selects a coordinate where it makes its move.

5 **How To Play:**

*System Requirements:*

1. Java Enabled Mobile Phone
2. MIDP 2.0 Required
3. CLDC 1.1 or above

*Game Controls:*

1. In the 3D view, use 4 to rotate anticlockwise and 6 to rotate clockwise.
2. Use 0 to change view mode from 3D to 2D to 4 layered classical view.
3. In the 4 layered classical you can navigate across layer using 2 or 8. To get to a higher layer use 2 and to get to a lower layer use 8.
4 Within a layer you can move by using 9-for right, 7-for left, 1-for up, 3-for down.

5 Use 5 to select a square.

Conclusion:
The project gave us one of the best programming experience we have had till now. It helped us learn practical application of java and working on multiple-platform. It also helped us to work in modules where different people were working on different parts of a project. There are many features which can be added to our project which will make it lot better. We can improve on the AI. There is also a scope of improvement on the graphical front though it depends on the user experience during gaming.