

..ㅋㅋㅋ

...

. ~~~

ㅋㅋㅋ

..

SK-VM

가

CP

.

..

가

..

..

.

..^^;

.

가

가

. --;

가

.

:

int len = is.read(buf);

buf

..

buf

가

.

...

:

bodyData = new byte[size];

int readn = 0;

while (readn < size) {

readn += in.read(bodyData, readn, bodyData.length - readn);

}

size 가 528byte

가

..

,

. ^^;

.

가

.

2

..



가
가
가

CSound.java

```
package Cube;

import com.skt.m.*;
import java.io.*;
import java.util.*;

public class CSound
{
    public final static int SOUND_NUM = 1;    //
    public final static int STOP = 10;

    AudioClip clip;
    byte buffer[][] = new byte[SOUND_NUM][];    //

    //
    public CSound()
    {
        try {
            clip = AudioSystem.getAudioClip("mmf");
            for (int i=0; i<SOUND_NUM; i++) {
                InputStream is =
                getClass().getResourceAsStream("/Cube/snd/sound" + i + ".mmf");
                buffer[i] = new byte[is.available()];
                is.read(buffer[i]);
            }
        } catch (Exception ex) {
        }
    }

    //
    // playSound(    )
    // playSound(STOP) -
    public synchronized void playSound(int num) {
        try {
            if(num != STOP){
```

```

        clip.open(buffer[num], 0, buffer[num].length);
        clip.play();
    }else{
        clip.stop();
    }
} catch (Exception ex) {
}finally {
    try {
        clip.close();
    }catch(Exception ex2) {
    }
}
}
}

```

CubeEngine.java

```

package Cube;

import javax.microedition.lcdui.*;
import java.util.*;
import java.io.*;
import com.skt.m.*;

public class CubeEngine
{
    // mode
    public final static int LOGO = 0;
    public final static int TITLE = 1;
    public final static int LEVEL = 2;
    public final static int MISSION = 3;
    public final static int GAME = 4;
    public final static int RUNMENU = 5;

    //
    public final static int KEY_UP = 141;
    public final static int KEY_LEFT = 142;
    public final static int KEY_RIGHT = 145;
    public final static int KEY_DOWN = 146;
    public final static int SOFT_LEFT = 129;
    public final static int SOFT_RIGHT = 131;

    //
    public int cellsize;
    public int xySize;
    public int cx, cy;
    // x, y
    public int menuSwitch = 1;
    //

```

```

        public int                levelSwitch = 1;
        public int                runSwitch = 1;
        private int               stageNumber;                //

        private int               mode;
        //
        private int               score;
        static int                swap;                        // cell

        static int[][] puzzle_num = new int[4][4];           // ( )

        static int[][] show_puzzle = new int[4][4];          //
        String[] stage = new String[3];                      // Stage

        //
        private CubeCube cube;
        private CubeCanvas canvas;                            //
Canvas
        private CSound sound;

        // Constructor
        public CubeEngine(CubeCanvas canvas)
        {
            this.canvas = canvas;
            sound = new CSound();

            stage[0] = new String("111112110");
        //
            stage[1] = new String("111112110");
            stage[2] = new String("111112110");
        /*
            stage[0] = new String("111212110");
        //
            stage[1] = new String("111324110");
            stage[2] = new String("131424130");
        */
        }

        // GET & SET method
        public int getMode()
        {
            return mode;
        }

        public void setMode(int mode)
        {

```

```

        this.mode = mode;
    }

    public int getStageNumber()
    {
        return stageNumber;
    }

    public void setStageNumber(int num)
    {
        stageNumber = num;
    }

    public int[][] getShowPuzzle()
    {
        return show_puzzle;
    }

    public int[][] getPuzzleNum()
    {
        return puzzle_num;
    }

    public void setCube(CubeCube cube)
    {
        this.cube = cube;
    }

    public void setPuzzle(int stageNum)
    {
        xySize = 3;
        cellsize=24;

        switch (stageNum)
        {
            case 1:
                setShowNum(3,0);
                setPuzzleNum(3,0);    //      ,
                break;
            case 2:
                setShowNum(3,1);
                setPuzzleNum(3,1);
                break;
            case 3:
                setShowNum(3,2);
                setPuzzleNum(3,2);
                break;
        }
    }

```

```

    }

    public void setShowNum(int size, int level)
    {
        int index=0;
        for(int i=0; i<size; i++){
            for(int j=0; j<size; j++){
                show_puzzle[i][j]
Integer.parseInt(stage[level].substring(index++,index));
            }
        }
    }

    public void setPuzzleNum(int size, int level)
    {
        int random = 0;
        random = getRandomInt(8);           // 0-7
        int ran = 1;
        for(int i=0; i<size; i++){
            for(int j=0; j<size; j++){
                if (random > size*size - 1) random = 0;
                ran
Integer.parseInt(stage[level].substring(random++,random));
                if (ran==0) {cx=i; cy=j;}
                puzzle_num[i][j] = ran;
            }
        }
    }

    static int getRandomInt(int limit) {
        Random rnd = new Random();
        int number = rnd.nextInt();
        number = (number >>> 16) & 0xffff;
        number /= (0xffff/ limit);
        return number;
    }

    public void showPuzzle()
    {
        mode = MISSION;
        canvas.drawShowPuzzle(stageNumber);
    }

    public void gameStart()
    {
        System.gc();
        canvas.paintCell();
    }

```

```

public void keyPress(int key)
{
    switch (mode)
    {
        case LOGO:                                // Logo
        case TITLE:
            keyPressInTitle(key);
            break;

        case LEVEL:                                // Show Level
            keyPressInLevel(key);
            break;

        case MISSION:                             // ShowPuzzle
            keyPressInShowPuzzle(key);
            break;

        case GAME:                                // In Game
            keyPressInGame(key);
            break;

        case RUNMENU:                             // Game Menu
            keyPressInRunMenu(key);
            break;
    }
}

```

```

public void keyPressInTitle(int key)
{
    if(mode == LOGO)
    {
        canvas.showMainMenu();
        mode = TITLE;
        return;
    }
    if ((menuSwitch != 1) && (key == KEY_UP))    //
    {
        menuSwitch - - ;
        canvas.showMainMenu();
        sound.playSound(0);
    }
    //
    else if ((menuSwitch != 5) && (key == KEY_DOWN)) //
    {
        menuSwitch++;
        canvas.showMainMenu();
    }
}

```

```

        sound.playSound(0);
    //
    }
}

public void selectMenu()
{
    switch (menuSwitch)
    {
        case 1:
            canvas.drawLevelMenu();
            mode = LEVEL;
            break;
        case 2:
            // (Instruction)
            //
            break;
        case 3:
            // (Credits)
            //
            break;
        case 4:
            // (Settings)
            //
            break;
        case 5:
            cube.gameEnd();
            break;
    }
}

public void keyPressInLevel(int key)
{
    if ((levelSwitch != 1) && (key == KEY_UP))
    {
        levelSwitch - -;
        canvas.drawLevelMenu();
        sound.playSound(0);
    }
}

```



```

    }
    else if ((levelSwitch != 3) && (key == KEY_DOWN)) //
    {
        levelSwitch++;
        canvas.drawLevelMenu();
        sound.playSound(0);
    }
}

public void selectLev()
{
    canvas.times = 0;
    switch (levelSwitch) // 가 ..
    {
        case 1: // Level 1 .
            setStageNumber(1);
            setPuzzle(1);
            showPuzzle();
            break;

        case 2: // 가 .
            break;

        case 3: // 가 .
            break;
    }
}

public void keyPressInShowPuzzle(int key)
{
    if (key == SOFT_LEFT) //
    {
        mode = GAME;
        gameStart();
    }
}

public void keyPressInGame(int key)
{
    if(key == SOFT_LEFT){
        canvas.timego = false;
        showPuzzle(); //

    }else{

```

```

switch (key)
{
    case KEY_UP: //
        if (cy > 0) {
            swap =
puzzle_num[cx][cy];
            puzzle_num[cx][cy] =
puzzle_num[cx][cy - 1];
            cy - -;
            puzzle_num[cx][cy] =
swap;
            canvas.drawMove(cx, cy,
cx, cy+1, cellsize, puzzle_num[cx][cy+1]);
        }
        break;

    case KEY_DOWN: //
        if (cy < xySize - 1) {
            swap =
puzzle_num[cx][cy];
            puzzle_num[cx][cy] =
puzzle_num[cx][cy+1];
            cy++;
            puzzle_num[cx][cy] =
swap;
            canvas.drawMove(cx, cy,
cx, cy - 1, cellsize, puzzle_num[cx][cy - 1]);
        }
        break;

    case KEY_LEFT: //
        if (cx > 0) {
            swap =
puzzle_num[cx][cy];
            puzzle_num[cx][cy] =
puzzle_num[cx - 1][cy];
            cx - -;
            puzzle_num[cx][cy] =
swap;
            canvas.drawMove(cx, cy,
cx+1, cy, cellsize, puzzle_num[cx+1][cy]);
        }
        break;

    case KEY_RIGHT: //
        if (cx < xySize - 1) {
            swap =
puzzle_num[cx][cy];
            puzzle_num[cx][cy] =
puzzle_num[cx+1][cy];

```

```

                                cx++;
                                puzzle_num[cx][cy] =
swap;
                                canvas.drawMove(cx, cy,
cx-1, cy, cellsize, puzzle_num[cx-1][cy]);
                                }
                                break;
                                }
                                }
                                }

public void keyPressInRunMenu(int key)
{
    if ((runSwitch != 1) && (key == KEY_UP)) //
    {
        runSwitch--;
        canvas.runMenu();
    }
    else if((runSwitch != 5) && (key == KEY_DOWN)) //
    {
        runSwitch++;
        canvas.runMenu();
    }
}
}

```

CubeCube.java

```

package Cube;

import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import java.io.*;
import com.skt.m.*;

public class CubeCube extends MIDlet
{
    public CubeCanvas canvas;
    static Graphics g;
    Display display;

    public CubeCube()
    {
        canvas = new CubeCanvas(this);
        Device.setBacklightEnabled(true); // 가
        가 ( true )
        Device.enableRestoreLCD(true); //
        . SUSPEND
    }
}

```

```

        loadLogo();
    }

    protected void startApp()
    {
        display = Display.getDisplay(this);
        canvas.active = true;
        display.setCurrent(canvas);
        BackLight.on(0); // On
    }

    protected void pauseApp()
    {
        canvas.active = false;
    }

    public void resumeApp()
    {
    }

    public void destroyApp(boolean uc)
    {
        display.setCurrent(null);
        notifyDestroyed();
    }

    public void loadLogo()
    {
        try{
            canvas.logo = Image.createImage("/Cube/img/logo.png");
        }catch(IOException e){
            e.printStackTrace();
        }
    }

    public void gameEnd()
    {
        destroyApp(false);
    }
}

```

CubeCanvas.java

```
package Cube;
```

```

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.*;

public class CubeCanvas extends Canvas implements Runnable, CommandListener
{
    //
    public final static int SCREEN_WIDTH      = 128;
    public final static int SCREEN_LENGTH    = 143;

    //
    static Image title, mt1, mt2, mt3, mt4, mcur, board, clear1, clear2, clear3, bt;

    static Image cell2_1, cell2_2, cell2_3, cell2_4, cursor, logo;

    //
    private CubeCube      cube;
    public  CubeEngine    engine;          //

    //
    public boolean        active = true;
    public boolean        init = true;
    public boolean        timego;          //
    On - Off .
    int[][] puzzle_num = new int[4][4];    //

    int[][] show_puzzle = new int[4][4];   //
    public int            moveCount;        //

    public int            sizeMAX;          //
    (3x3, 4x4)
    public int            times;            //

    private int           tc;
    //      repaint
    private Thread        thread;
    static Graphics       g;

    private Command cmd1 = new Command("START", 2, 0);
    private Command cmd2 = new Command("GO!", 4, 0);
    private Command cmd3 = new Command("MAP", 2, 0);
    private Command cmd4 = new Command("MENU", 8, 1);
    private Command cmd5 = new Command("BACK", 2, 0);
    private Command cmd6 = new Command("SELECT", 4, 1);
    private Command cmd7 = new Command("OK!", 4, 1);
    private Command cmd8 = new Command("", 3, 0);          // NULL

```

```

// Constructor
public CubeCanvas(CubeCube cube)
{
    this.cube = cube;
    engine = new CubeEngine(this);
    engine.setCube(cube);
    loadImage();
    thread = new Thread(this);
    thread.start();
}

protected void paint(Graphics g) {
    this.g = g;
    if(init){
        g.drawImage(logo, 0, 22, 0);
        init = false;
    }
}

//
public void showMainMenu(){
    drawBox(20, 25, 88, 94);
    g.setColor(255, 255, 255);
    g.drawString("Start Game", 64, 38, 16|1);
    g.drawString("Instruction", 64, 52, 16|1);
    g.drawString("Credits", 64, 66, 16|1);
    g.drawString("Settings", 64, 80, 16|1);
    g.drawString("Exit", 64, 94, 16|1);
    drawCursor();
    addCommand(cmd6);
    setCommandListener(this);
    repaint();
    serviceRepaints();
}

//
public void drawCursor()
{
    switch(engine.menuSwitch)
    {
        case 1:
            g.drawImage(mcur, 27, 43, 0);
            g.drawImage(mcur, 94, 43, 0);
            break;
        case 2:
            g.drawImage(mcur, 25, 43+(14*1), 0);

```

```

        g.drawImage(mcur, 97, 43+(14*1), 0);
        break;
        case 3:
            g.drawImage(mcur, 35, 43+(14*2), 0);
            g.drawImage(mcur, 86, 43+(14*2), 0);
        break;
        case 4:
            g.drawImage(mcur, 32, 43+(14*3), 0);
            g.drawImage(mcur, 89, 43+(14*3), 0);
        break;
        case 5:
            g.drawImage(mcur, 42, 43+(14*4), 0);
            g.drawImage(mcur, 78, 43+(14*4), 0);
        break;
    }
}

//
public void drawBox(int x, int y, int w, int h) {
    g.setColor(0, 0, 0);
    g.fillRect(x, y, w, h);
    g.setColor(0, 109, 85);
    g.fillRect(x+3, y+3, w-6, h-6);
    g.setColor(255, 219, 0);
    g.drawRect(x+1, y+1, w-3, h-3);

    g.drawImage(mt1, x, y, 0);
    g.drawImage(mt2, x+w-15, y, 0);
    g.drawImage(mt3, x, y+h-16, 0);
    g.drawImage(mt4, x+w-15, y+h-16, 0);
}

//
public void loadImage()
{
    try
    {
        mt1 = Image.createImage("/Cube/img/m01.png");
        mt2 = Image.createImage("/Cube/img/m02.png");
        mt3 = Image.createImage("/Cube/img/m03.png");
        mt4 = Image.createImage("/Cube/img/m04.png");
        mcur = Image.createImage("/Cube/img/mcur.png");
        board = Image.createImage("/Cube/img/board.png");
        clear1 = Image.createImage("/Cube/img/clear1.png");
        clear2 = Image.createImage("/Cube/img/clear2.png");
        clear3 = Image.createImage("/Cube/img/clear3.png");
        bt = Image.createImage("/Cube/img/bt.png");
        cell2_1 = Image.createImage("/Cube/img/cell2-1.png");
        cell2_2 = Image.createImage("/Cube/img/cell2-2.png");
    }
}

```

```

        cell2_3 = Image.createImage("/Cube/img/cell2-3.png");
        cell2_4 = Image.createImage("/Cube/img/cell2-4.png");
        cursor = Image.createImage("/Cube/img/cursor.png");

        } catch(IOException ex){
            System.out.println("File not find");
        }
    }

    //
    public void drawLevelMenu()
    {
        drawBox(25, 35, 78, 72);
        g.setColor(255, 255, 255);
        g.drawString("[Level 1]", 65, 48, 16|1);
        g.drawString("[Level 2]", 65, 64, 16|1);
        g.drawString("[Level 3]", 65, 80, 16|1);
        drawCursor2();
        addCommand(cmd8);
        addCommand(cmd6);
        setCommandListener(this);
        repaint();
        serviceRepaints();
    }

    //
    public void drawCursor2()
    {
        switch(engine.levelSwitch)
        {
            case 1:
                g.drawImage(mcur, 30, 53+(16*0), 0);
                g.drawImage(mcur, 94, 53+(16*0), 0);
                break;
            case 2:
                g.drawImage(mcur, 30, 53+(16*1), 0);
                g.drawImage(mcur, 94, 53+(16*1), 0);
                break;
            case 3:
                g.drawImage(mcur, 30, 53+(16*2), 0);
                g.drawImage(mcur, 94, 53+(16*2), 0);
                break;
        }
    }

    //
    public void drawBar()
    {

```



```

        g.setColor(252, 218, 4);
        g.drawLine(97, 0, 97, 108);
        g.drawImage(board, 98, 0, 0);
        g.setColor(220, 255, 172);
        g.fillRect(109, 20, 10, 9);
        g.setColor(0, 0, 0);
        g.drawString(""+(engine.getStageNumber()), 118, 18,
Graphics.TOP|Graphics.RIGHT);
        repaint();
    }

    //
    public void drawShowPuzzle(int stageNum)
    {
        drawBar();

        drawBackImg();

        g.setColor(0, 0, 0);
        timer(times);
        drawCount();

        if(engine.getMode() != engine.GAME)
        {
            removeCommand(cmd6);
            removeCommand(cmd8);
            addCommand(cmd1);
            setCommandListener(this);
        }

        drawBox(18, 8, 93, 115);

        show_puzzle = engine.getShowPuzzle();

        sizeMAX = 3;
        for(int i=0; i < sizeMAX; i++){
            for(int j=0; j < sizeMAX; j++){
                drawCell(25 * i +28 , 25 * j +27, show_puzzle[j][i]);
            }
        }

        g.setColor(255, 255, 255);
        g.drawString("YOUR MISSION", 28, 14, 0);
        g.drawString("STAGE "+stageNum, 43, 103, 0);
        repaint();
        serviceRepaints();
    }

    //
    public void drawBackImg()

```

```

{
    g.drawImage(bt, 0, 109, 0);

    switch(engine.getStageNumber())
    {
        case 1:
            g.drawImage(clear1, 0, 0, 0);
            break;

        case 2:
            g.drawImage(clear2, 0, 0, 0);
            break;

        case 3:
            g.drawImage(clear3, 0, 0, 0);
            break;
    }
}

//
public void drawMove(int cX, int cY, int oX, int oY, int size, int swap)
{
    g.drawImage(cursor, cX*(size+1)+12, cY*(size+1)+17, 0);
    drawCell(oX*(size+1)+12, oY*(size+1)+17, swap);
    drawCount();
    repaint();
}

//
public void paintCell()
{
    drawBar();
    //
    drawBackImg();
    //

    g.setColor(0, 0, 0);
    g.drawRect(9, 14, 79, 79); //

    g.setColor(255, 255, 0);
    g.fillRect(10, 15, 78, 78);

    puzzle_num = engine.getPuzzleNum();
    timego = true;
    timer(times);
    drawCount();

    for(int i=0; i < sizeMAX; i++){ // 3X3

```

```

        for(int j=0; j < sizeMAX; j++){
            drawCell(25 * i +12, 25 * j +17, puzzle_num[i][j]);
        }
    }
    repaint();
}

//
public void drawCell(int x, int y, int num)
{
    switch(num)
    {
        case 1:
            g.drawImage(cell2_1, x, y, 0);
            break;
        case 2:
            g.drawImage(cell2_2, x, y, 0);
            break;
        case 3:
            g.drawImage(cell2_3, x, y, 0);
            break;
        case 4:
            g.drawImage(cell2_4, x, y, 0);
            break;
        case 0:
            g.drawImage(cursor, x, y, 0);
            break;
    }
    repaint();
}

//
public void timer(int time)
{
    g.setColor(220, 255, 172);
    g.fillRect(103, 55, 20, 10);
    g.setColor(0, 0, 0);
    g.drawString(""+(time), 124, 53, Graphics.TOP|Graphics.RIGHT);
}

//
public void drawCount()
{
    g.setColor(220, 255, 172);
    g.fillRect(103, 89, 20, 9);
    g.setColor(0, 0, 0);
    g.drawString(""+(moveCount++), 124, 87,
Graphics.TOP|Graphics.RIGHT);
}

```

```

//
public void runMenu()
{
    drawBox(20, 25, 88, 94);
    engine.setMode(engine.RUNMENU);
    g.setColor(255, 255, 255);
    g.drawString("Pause", 64, 38, 16|1);
    g.drawString("Resume", 64, 52, 16|1);
    g.drawString("Score", 64, 66, 16|1);
    g.drawString("Back to main", 64, 80, 16|1);
    g.drawString("Exit", 64, 94, 16|1);
    drawRunCursor();
}

//
public void drawRunCursor()
{
    switch(engine.runSwitch)
    {
        case 1:
            g.drawImage(mcur, 37, 43+(14*0), 0);
            g.drawImage(mcur, 84, 43+(14*0), 0);
            break;
        case 2:
            g.drawImage(mcur, 30, 43+(14*1), 0);
            g.drawImage(mcur, 90, 43+(14*1), 0);
            break;
        case 3:
            g.drawImage(mcur, 35, 43+(14*2), 0);
            g.drawImage(mcur, 85, 43+(14*2), 0);
            break;
        case 4:
            g.drawImage(mcur, 25, 43+(14*3), 0);
            g.drawImage(mcur, 96, 43+(14*3), 0);
            break;
        case 5:
            g.drawImage(mcur, 42, 43+(14*4), 0);
            g.drawImage(mcur, 78, 43+(14*4), 0);
            break;
    }
    repaint(20, 25, 88, 94);
}

public void run() {
    try {
        while(active)
        {

```

```

        thread.sleep(1000);
        if(timego == true){
            timer(times++);
        }
        repaint();
    }
} catch (InterruptedException e) {
}
}

public void commandAction(Command c, Displayable d)
{
    String sel = c.getLabel();
    if(sel.equals("SELECT"))
    {
        if(engine.getMode() == engine.TITLE)
        {
            engine.selectMenu();
        }
        else if(engine.getMode() == engine.LEVEL)
        {
            engine.selectLev();
        }
    }
    else if(sel.equals("START"))
    {
        removeCommand(cmd1);
        addCommand(cmd4);
        addCommand(cmd3);
        engine.keyPress(engine.SOFT_LEFT);
    }
    else if(sel.equals("MAP"))
    {
        removeCommand(cmd3);
        removeCommand(cmd4);
        addCommand(cmd2);           // GO!
        addCommand(cmd8);          // NULL
        setCommandListener(this);
        timego = false;
        drawShowPuzzle(engine.getStageNumber());
        engine.setMode(engine.MISSION);
    }
    else if(sel.equals("MENU"))
    {
        removeCommand(cmd3);
        removeCommand(cmd4);
        addCommand(cmd5);
        addCommand(cmd6);
        setCommandListener(this);
    }
}

```

```

        timego = false;
        runMenu();
    }
    else if(sel.equals("BACK"))
    {
        removeCommand(cmd5); // BACK
        removeCommand(cmd6); // SELECT
        addCommand(cmd3);
        addCommand(cmd4);
        engine.setMode(engine.GAME);
        paintCell();
    }
    else if(sel.equals("GO!"))
    {
        removeCommand(cmd2);
        removeCommand(cmd8);
        addCommand(cmd3);           // GO!
        addCommand(cmd4);           // NULL
        timego = true;
        engine.setMode(engine.GAME);
        paintCell();
    }
}

protected void keyPressed(int keyCode) {
    engine.keyPress(keyCode);
}
}

```

```

.                                     ———;                                     .
                                     Neoco                                     Neoco
                                     ~~~~~
                                     ..                                     ..

Neoco                               . ㄐㄐㄐ   Early Adopter   ..

:
:
:
: pinkred@hanafos.com
:
:
:

```