# 게임..ㅋㅋㅋ 내손안에 있소이다...

안녕하세요 핑크레드 입니다. 제가 군대를 갔다 오느라 강좌에 소홀했습니다.
국방의 의무를 하느라..절 아마 여자분이시라고 생각하신분은 아마 있으실려나..ㅋㅋㅋ

말투를 봐도 이것저것 봐도..남자죠..성격이 점 드러울뿐이죠..^^;
오늘은 언능 강좌를 올리느라..할말을 짧게 하겠습니다.

일반적으로 액션 게임 같은 경우에는 Thread 를 돌리면서 도대체 어떻게 하면 최적화되게.. 빠르게 할수 있을까 라는 생각을 하게 됩니다.

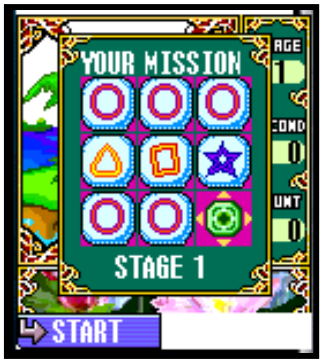
Run 문에 들어갈 것에 대해서 간략하게 소개 해 드리겠습니다. 도움이 되길 빕니다.

```
public void run()
{
    Thread.currentThread().setPriority(Thread.MAX_PRIORITY);

...
...
    repaint(x, y, width, height);
    serviceRepaints();
    Thread.yield();

}
```

보통 이런 구조가 가장 스피드하게 돌아가는 구조중의 하나입니다. 이것이 좋다는 것은 아니고 단지 하나일뿐입니다. 나중에 어 해보니 느리던데요..라고 말씀하신다면.ㅡㅡ; 할말 없습니다. 오늘은 여기서 끝내고 바로 들어갑니다.




이번에는 스테이지의 미션제시 화면을 생성하고, 본 게임의 스테이지 화면까지 만들어 보도록 하겠습니다. 첨부된 예제 소스를 잘 분석하셔서 직접 작성해 보시기를 바랍니다. 미션제시 화면의 확인 후엔 바로 본게임 스테이지 화면이 나오며, 아직은 본게임에서 키입력은 되지 않습니다. 이번 단계에서는 본격적인 메뉴 네비게이션을 위해서 Command 버튼들도 추가 해서 사용하도록 하겠습니다.

CSound.java

```
package Cube;

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

public class CSound
{
	public final static int SOUND_NUM   = 1;	// 사운드 파일의 개수.
	public final static int STOP		= 10;

	AudioClip clip;
	byte buffer[][] = new byte[SOUND_NUM][];		// 사운드 파일이 많아질 경우를 대비해 배열로 잡음.

	//사운드 로딩 및 생성
	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) {
		}
	}

	// 사운드 플레이
	// playSound(번호) 형식으로 호출하면 해당번호 사운드 재생
	// playSound(STOP) - 사운드 출력정지
	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 부분 상수들.
	public final static int LOGO		= 0;
	public final static int TITLE		= 1;
	public final static int LEVEL		= 2;
	public final static int MISSION		= 3;

	// 키입력부분 상수들.
	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;


	// 멤버 변수
	public int cellsize;
	public int xySize;
	public int cx, cy;
	// 커서의 x, y 값
	public int menuSwitch = 1;		// 메뉴 선택을 위해서 쓰이는 모드변수
	public int levelSwitch = 1;
	public int runSwitch = 1;
	private int stageNumber;		// 현재 스테이지의 넘버
	private int mode;
	// 화면에 따른 키이벤트를 구분하기 위해서 쓰이는 모드변수.
	static int swap;			// cell 의 이동시 값 치환을 위한 임시변수
	static int[][] puzzle_num = new int[4][4];  // 퍼즐의 각 픽셀(사각형)에 들어갈 숫자배열
	static int[][] show_puzzle = new int[4][4];  // 미션제시 퍼즐 숫자배열
	String[] stage = new String[3];		//모든 Stage 정보를 담는 배열
```

```
	// 참조클래스 선언
	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("131424130");
		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);	//  사이즈, 레벨
			break;
			case 2:
				setShowNum(3,1);
				setPuzzleNum(3,1);
			break;
			case 3:
				setShowNum(3,2);
				setPuzzleNum(3,2);
			break;
			case 4:
				setShowNum(3,3);
				setPuzzleNum(3,3);
			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 랜덤출력
		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;}                    //  커서
위치 초기화 하기.
                                    puzzle_num[i][j] = ran;
                            }
                    }
            }

            static int getRandomInt(int limit) {          // 랜덤하게 숫자 뽑기. limit 이내.
                    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;

                    }
            }
```

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


	public void selectMenu()
	{
		switch (menuSwitch)    // 현재의 선택레벨을 가지고 감..
		{
			case 1:
				canvas.drawLevelMenu();           // 레벨메뉴 보이
기.
				mode = LEVEL;
			break;

			case 2:
				// 게임설명 (Instruction)
				// 이부분은 레벨메뉴를 띄우는 것과 동일한 방식으로
작성 됨
			break;

			case 3:
				// 저작권 정보 (Credits)
				// 이부분은 레벨메뉴를 띄우는 것과 동일한 방식으로
작성 됨
			break;

			case 4:
```

```
                                        // 환경설정 (Settings)
                                        // 이부분은 레벨메뉴를 띄우는 것과 동일한 방식으로
작성 됨
                                break;

                                case 5:
                                        cube.gameEnd();             // 게임종료 (Exit)
                                break;
                        }
                }



                public void keyPressInLevel(int key)
                {
                        if ((levelSwitch != 1) && (key == KEY_UP))    // 위방향
                        {
                                levelSwitch--;
                                canvas.drawLevelMenu();
                                sound.playSound(0);
                        }
                        else if ((levelSwitch != 3) && (key == KEY_DOWN))  // 아래방향
                        {
                                levelSwitch++;
                                canvas.drawLevelMenu();
                                sound.playSound(0);
                        }
                }


                public void selectLev()
                {
                        switch (levelSwitch)    // 현재의 선택레벨을 가지고 감.
                        {
                                case 1:                                 // Level 1 선택.
                                        setStageNumber(1);
                                        setPuzzle(1);
                                        showPuzzle();
                                break;

                                case 2:
                                        // 추가형식.
                                break;

                                case 3:
                                        // 추가형식.
                                break;
                        }
                }
```

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

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);		// 키가 눌려질때 마다 백라이트가 들어오는 기능설정 (기본적으로 true 로 설정)
		Device.enableRestoreLCD(true);		// 통화후 이전화면 복구설정. SUSPEND 모드에서 돌아올때 화면복구를 위해
		loadLogo();
	}

	protected void startApp()
    {
		display = Display.getDisplay(this);
	    canvas.active = true;
		display.setCurrent(canvas);
		BackLight.on(0);			// 백라이트 On 으로 설정. 파라미터 0 이면 무한대 켜짐.
    }

    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
{
        // 상수선언
        public final static int SCREEN_WIDTH          = 128;
        public final static int SCREEN_LENGTH         = 143;

        // 이미지 변수선언
        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;
```

```
	// 참조클래스 선언
	private CubeCube		cube;
	public  CubeEngine			engine;


	// 변수선언
	public boolean		active = true;
	public boolean		init = true;
	public boolean		timego;				// 타이머의 진행
을 조절하는 On-Off 변수.
	int[][] puzzle_num = new int[4][4];		// 게임중 퍼즐을 저장하는
배열
	int[][] show_puzzle = new int[4][4];	// 미션 퍼즐을 저장하는 배열
	public int				moveCount;			// 커서
의 움직인 횟수를 카운드하는 변수.
	public int				sizeMAX;		// 셀의 크기를 결
정 (3x3, 4x4)
	public int				times;				// 타이
머 누적시간 변수
	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;
        }
    }


    // 메인메뉴 띄우기
    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();
    }


    // 메인메뉴 선택커서표시
    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;
        }
    }
```

```
// 넘겨준 크기값으로 뒷배경사각형을 그리는 부분
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);
}

// 이미지 로딩
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");
	}
}


//레벨선택 메뉴 그리기
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();
        }

        // 레벨메뉴 선택커서
        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;
                }
        }

        // 화면오른편 게이지보드
        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();
        }

        // 미션제시화면의 표시
        public void drawShowPuzzle(int stageNum)
        {
                drawBar();
                                        // 오른편 점수표 그리기
                drawBackImg();
```

```
                                        // 배경이미지 그리기
        g.setColor(0, 0, 0);
        drawCount();
        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();
    }

    // 뒷배경 이미지 그리기
    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;
        }
    }



    // 본게임 화면의 표시
    public void paintCell()
```

```
	{
		drawBar();
				// 오른편 점수표 그리기
		drawBackImg();
		 // 배경이미지 그리기

		g.setColor(0, 0, 0);
		g.drawRect(9, 14, 79, 79);                                          //  퍼즐
바깥의 테두리
		g.setColor(255, 255, 0);
		g.fillRect(10, 15, 78, 78);

		puzzle_num = engine.getPuzzleNum();
		drawCount();

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

	// 정해준 위치에 크기에 따라 셀을 그림.
	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();
	}

	// 움직인 이동횟수의 표시
	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 run() {
          try {
                              while(active)
                              {
                              thread.sleep(200);
                                        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);
                    }


          }

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

프로그램을 제 방식으로 손댈 수 있지만 여러분의 학습상 도움이 될 것 같아서 일단은 접었습니다. 사실은 프로그램 고치기 무지 귀찮더군요 ㅡㅡ; 그냥 올립니다. 이 소스는 Neoco 님의 저작권이 달린 소스이니 소스를 사용시에는 Neoco 님의 허락을 받으셔야 할겁니다.
강좌는 계속 이어집니다~~~~~

그럼 즐거운 하루 되십시오..핑크레드..
참고로 Neoco 님 또한 소심한 면이 있습니다. ㅋㅋㅋ 자칭 Early Adopter 라고 하시네요..

닉네임 : 핑크레드
성격 : 다소 소심, 다소 삐짐, 다소 이중인격, 다소 수다쟁이
이메일 : pinkred@hanafos.com

추신 : 강좌를 다른 사이트에 올리시는 것은 자유입니다..단지 저한테 메일 한 통 이라도 보내주시고 올리시면 감사하겠습니다. 언제나 좋은 하루 되십시오.