게임..ㅋㅋㅋ 내손안에 있소이다...

ㅋㅋㅋ 소스가 있으니..강좌 하기가 무지 편하군요. 특별히 이야기도 필요없고…

이번에 제가 몇번째 강좌인지 기억은 가물하지만 잘못 알려드린 부분이 있어서 내용을 수정합니다. 해당 강좌의 내용도 수정해야 하는데..언능 찾아야죠…

일단 내용인 즉슨 setClip 의 문제 였습니다. 제가 알아보기로는 이 함수를 사용하면 기존의 단말기에서 repaint 시에 많은 이점이 있으므로 사용을 권하고 했는데..문제는 제가 테스트 예제를 돌리다가 문득.. 이점의 의심되는바 검증을 하다가 알아낸 부분입니다.

해본 결과 의미 없다가 정답이었습니다. SetClip 의 의미는 그냥 단순히 영역을 설정하는 것 외에는 크게 의미가 없다는 것이었습니다. 그럼 어떻게 하면 더 효율적일까..라는 부분에서 바로 repaint(x, y, width, height)함수를 이용하는 것이었습니다. 이 함수는 실제적으로 단말기 LCD 화면에 해당 부분만 그려주기 때문에 훨씬 더 빠르다는 것은 알고 있지만..이러한 부분을 힘들게 해당 좌표 영역을 전부 설정해주어어야 하기 때문에 안좋은 점이 있어서 setClip 를 이용했지만..결과적으로 setClip 의 의미가 없어졌기 때문에 기존의 repaint(x, y, width, height)를 이용하시는 편이 더 좋다고 생각합니다. 그런데 테스트 중에 아주 기묘한 결과에 제가 약간 당황했습니다. 문제의 발생은 바로 Thread.sleep()이였습니다. 보통 우리는 Thread 사용시에 루프문에 sleep 을 줍니다. 이유중에 하나가 다른 이벤트를 받기 위해서지요. 그런데 문제는 여기서 발생했습니다. Repaint(), serviceReapints(), reapint(x, y, width, height) 등의 모든 테스트를 하는데 결과가 거의 동일하게 나왔습니다. 그 원인이 바로 sleep 이였습니다. Thread 가 sleep 인 동안에 모든 이벤트 들이 끝나기 때문에 여러가지 빠르게 수행하는 것들이 의미가 사라지는 것이였습니다. **결과적으로 sleep 의 효과로 인하여 최적화 속도의 모든 작업들이 무의미 해지는 부분이였습니다.** 아주 중요한 발견이라고 할수 있습니다. Sleep 을 대처 할수 있는 Thread.yield()로 수행 하여 이러한 부분들의 문제를 해결 할 수가 있었습니다. 참고로 Thread.yield()를 수행하면 순간적으로 다른 이벤트를 처리하기 때문에 기존의 최적화 속도에 공헌을 하지만..문제는 빠른 화면을 쓸 때..화면을 전부 그리는 것에 대해서 단말기 마다 보장을 하지 못하더군요.. 모든 라인을 하나씩 그리는 테스트를 했는데..yield()를 사용하면 중간에 그리지 못하는 라인도 있더군요.. 그래도 단말기 마다 다르지만 속도는 기존의 것보다 최소 1.5 배 이상 나오더군요.. 실질적인 테스트 데이터는 공개해 드리지 못합니다. 그러한 이유는 위의 상황과 같이 어처구니 없는 부분들이 많기 때문에..이러한 결과가 나오지요.. 혼자서 나온 데이터를 열심히 분석해본 결과..얻은 것입니다. 데이터가 없으므로 사기다 라고 하시는 분들은 기존의 방식데로 하셔도 무방합니다. 전 단지 여러 폰의 테스트 결과 나온 것을 가지고 이것저것 생각하다가 얻은 결론입니다.

또한 기존의 데이터를 착실히 적어놓고 시작은 하였으나..결과의 황당함에 삭제를 시켜버리고 나름대로..정의 하느라 자료가 없습니다. 믿거나 말거나..

그러면 또..게임 소스 진행을 해야지용.. 제가 저번주까지..계속 올릴수 있었지만.. 갑자기 바뻐짐에 따라서..아시겠죠...변명입니다. 이번강좌는 레벨 화면 추가 입니다. 아직 게임은 아니고 초기에 시작할 때 필요한 정보를 얻는 것이라고 할수 있죠.. 간단하지만 필요한 부분들이죠..

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;

	// 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				mode;
	// 화면에 따른 키이벤트를 구분하기 위해서 쓰이는 모드변수.
	public int				menuSwitch = 1;			//	메뉴
선택을 위해서 쓰이는 모드변수
	public int				levelSwitch = 1;
	private CubeCube cube;
```

```
	private CubeCanvas	canvas;					//
Canvas
	private CSound	sound;


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


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

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

			case LEVEL:				// Level
				keyPressInLevel(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);
	// 사운드파일 번호
		}
```

```
                else if ((menuSwitch != 5) && (key == KEY_DOWN))  // 아래방향
                {
                        menuSwitch++;
                        canvas.showMainMenu();
                        sound.playSound(0);
        // 사운드파일 번호
                }
                else if (key == SOFT_RIGHT)
                {
                        switch (menuSwitch)
                        {
                                case 1:
                                        canvas.drawLevelMenu();          //  레벨
메뉴 보이기.
                                        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);
                }
                else if (key == SOFT_RIGHT)                     // 오른쪽소프트 확인키
                {
                        switch (levelSwitch)    // 현재의 선택레벨을 가지고 감.
                        {
                                case 1:             // Level 1 선택..
                                        // 추가형식.
                                break;

                                case 2:
                                        // 추가형식.
                                break;

                                case 3:
                                        // 추가형식.
                                break;
                        }
                }

        }

}
```

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 백
라이트가 들어오는 기능설정 (기본적으로 true 로 설정)
```

```
                Device.enableRestoreLCD(true);              // 통화후 이전화면 복구설
정. 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
{
	// 상수선언
	public final static int SCREEN_LENGTH	= 128;
	public final static int SCREEN_WIDTH	= 143;

	// 이미지 변수선언
	static Image title, mt1, mt2, mt3, mt4, mcur;

	// 참조클래스 선언
	private CubeCube	cube;
	public  CubeEngine		engine;		// 게임의 로직부분을 담당
하는 클래스를 하나더 생성합니다.

	public boolean		active = true;
	public boolean		init = true;
	private Thread		thread;
	static Graphics g;
	static Image logo;


	// Constructor
	public CubeCanvas(CubeCube cube)
	{
		this.cube = cube;
		engine = new CubeEngine(this);
		engine.setCube(cube);
		loadImage();	// 이부분을 engine 쪽으로 옮기면, 분산로딩 가능함.
		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();
	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");
		}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();
		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;
                }
        }

    public void run() {
        try {
                        while(active)
                        {
                        thread.sleep(200);
                        }
                } catch(InterruptedException e) {
            }
        }


        // Canvas 는 화면의 이미지를 담당하여 그리는 클래스이며, Engine 은 게임내부의
로직을
        // 담당하는 클래스로 놓아두면, 역할의 분담이 가능하여 소스가 명료해 집니다.
        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.