게임..ㅋㅋㅋ 내손안에 있소이다...

안녕하세요~~!!! 핑크레드 입니당…
1 편이 너무 작어서..거진 형편이 없죠..그래서 빨랑 2 편을 내놓았습니다. 초보분들이 싱거워 하실가봐 ㅡㅡ; 소금을 점 그래도 쳐야지 맛이 나지 않을까..

2 편은 1 편에서 만든것에 기본 메뉴만 올리는 것입니다. 이것 또한 심심하죠..그래도..1 편보다는 먼가 있어보이잔아요... 이런 것이 바로 장수 늘리기와 질~~질..오래 끌기의 진수죠.

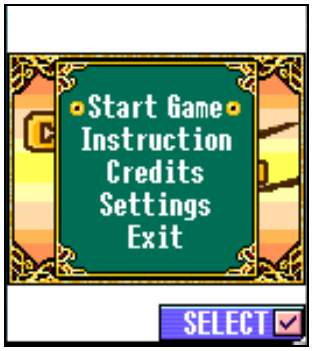
답답하더라도.. 참아내면서 한편씩 읽어 나가셔야 할 것 같음..ㅋㅋㅋ 다음편까지 기다리시기 힘드신 분은 그냥..참으셈.

요즘에는 PC 프로그래머들이 모바일 쪽으로 전향하시는 분이 점 있으신 것 같습니다. 그렇다 보니.. 모바일 프로그램을 PC 프로그램처럼 똑같이 짜시더군요 ㅡㅡ; 가끔씩 프로그램 점 봐달라고 하시는 분들이 있으신데 보면 이런 분들이 태반입니다. 모라구 할 수는 없죠. 경험이 없으니.. 수정은 해드리지만.. 전부 해드리기도 모합니다. 여기저기 글들을 보면 자바를 최적화하는 방법이 여기저기 있으니깐요.. 공부가 중요하다는 것이지요... 여튼 결과적으로는 열심히 하자네요..ㅋㅋㅋ 그리고 너무 오버해서..하지 마세요 ㅡㅡ; 가끔씩 코드를 보면 오버하시는 분들도 있는데.. 오버는 정신 건강에 해롭습니다.

이제는 할말도 거진 없습니다. 적지만..10 편정도의 강좌에 이런말 저런말을 쓰다보니..이제는 바닥났습니다. 나중에 새로운 것을 알아내면 적도록 하지요.. 아차..방금 기억이 났습니다. ㅋㅋㅋ 보통 많은 분들이 Thread 보다는 타이머를 많이 사용합니다. 왜냐하면 편리하기 때문이죠..하지만 Timer 와 Thread 는 점 다르죠.. Timer 가 내부적으로 Thread 로 만들어져 있습니다. 왜냐구요.. 생각해보심 이해가 되지요..그래서 용도에 맞게 쓰자는 것이 제 생각입니다. Timer 보다는 Thread 가 더 편하고 가볍지요.. 그러나 Timer 가 꼭 필요할 때도 있습니다. 보통 Timer 는 어느 일정 시간이 지나면 실행되기 때문에 그러한 용도를 쓰는 곳에서 사용하면 되고 Thread 는 언제나 똑 같은 일을 해야 하는 것에 사용하면 됩니다. 둘 다 같은 내용이라고요.. 예를 들어 화면을 계속 repaint()하는 것은 Timer 보다는 Thread 가 좋고 다이얼로그 창이 3 초 후에 자동으로 사라지게 할려면 Timer 가 좋다는 것이지요. 이해가 되셨는지요.

그럼 이제 본론으로 들어가서..

2 편에서는 1 편에서 만들어진 메뉴위에는 커서도 삽입하며, 상하 방향키에 따라 메뉴커서도 이동하도록 합니다. 그리고 CubeEngine 클래스가 등장합니다. 사용자의 키입력에 따라 커서를 이동시키고, 화면의 상태를 mode 값으로 분류하여 이벤트 키값을 적절한 메쏘드로 분리하도록 합니다.



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;
	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();
                }
        }

}
```

CubuCanvas.java

```
package Cube;

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


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

        static Image title, mt1, mt2, mt3, mt4, mcur;
```

```
        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);
                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();
                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");
                }
        }


    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) {     // Canvas 안의 API 메쏘드..
                engine.keyPress(keyCode);
        }


}
```

CubeEngine.java

```
package Cube;

import javax.microedition.lcdui.*;
import java.util.*;

public class CubeEngine
{
        // mode 부분 상수들.
        public final static int LOGO                    = 0;
        public final static int TITLE                   = 1;

        // 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;

        // 멤버 변수
```

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


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


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

		}
	}



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

}
```

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.