핑크레드의 실전 모바일 자바(SK-VM) 프로그래밍
(가칭 : 구루구루 춥)

안녕하세요 핑크레드 입니다.

게임 도움 강좌를 쓴지도 언..몇 달이 흐르고.. 무엇을 할까 고민했는데..
이넘의 회사가 자작 프로그램을 만들어도 팔아 줄생각을 안하니..중독성 게임을 만들었는데 약간 기획을 더 첨부해서 양념하고 맛점 보고 이것저것 하면 될 것 같은데..

회사에서 관심이 없어서 그냥 공개하기로 했습니다. 어차피 제가 만든 거고 공개해도 별 문제 없으리라 생각됩니다. 기존의 소스 코드를 최적화 하지는 못했고 디자인도 제가 일일이 하다보니 깨끗하지도 않지만 잼있는 겜이라고 생각합니다. 나중에 보시고 맘에 드시는 분 있으시면..추천 점 해주세요..저도 겜점 팔게..ㅎㅎㅎ

일단 해당 게임을 설명 드리겠습니다. 가칭으로 “구루구루 춥” 이라고 했습니다. 솔직히 다날이 영업은 진짜 잘한 것 같습니다. 요즘 모바일 게임도 CF찍고 광고 하고 이렇게 하지 않으면 장사가 안되니 그런걸 봐서 “고르고르 촙”은 성공작품 이라고 생각합니다. 이런류 중에서 이렇게 인기 좋은 것도 없으니깐요. 특히 SKVM인데도 불구하고..ㅎㅎㅎ

여튼 그런 것이 부러워서 가칭 “구루구루 춥”이라고 지었습니다. 내심..다날에서 사주었으면 한다는 간절한 바람도 있지만 소스도 공개하니 여러 CP업체분들은 한번 만들어 주시고..라이센스료나 조금 주시면 영광입니다. 솔직히 제가 무슨 라이센스나 있겠습니까..이름 한줄 넣어주세요..ㅜㅜ

만들 게임은 예전에 도스용 퍼즐 게임인 원제 “Torus” 라는 게임입니다. 제가 대학생 시절에 열심히 과친구들과 했던 게임으로 상당히 중독성이였었는데 모바일로는 없어서 한번 만들어 봤습니다. 속도도 잘나오고 별 문제 없어서 진행도 잘되었는데..이넘의 회사가 관심이 없더군요. 디자인은 도스용 게임이기 때문에 스크린샷으로 잡아서 만들었습니다. 제가 도통 디자인이 형편 없어서 고생점 했습니다. 그나마 도스용 게임이라도 있으니 디자인이 점 됬지 맨땅에 해딩할뻔 했습니다. 첨부로 해당 겜도 넣어드리니 보시고 평가해 주시면 감사하겠습니다.

해당 강좌는 꽤 나가야 할 것 같습니다. 한번에 되는 게임도 아니고..차근차근 하나씩 예전에 큐브큐브 하듯이 진행해야 할 듯 하네요..^^

제가 처음부터 이게임을 만들기위해서했던 모든 것들을 차근차근 해서 알려드리겠습니다.

많은 분들이 보시면서 이겜에 재미를 느꼈으면 하는 소망이 있습니다..^^

참고로 제가 기획자가 아니기 때문에 양식이나 형식이나 기타등등 미숙할수 있음.

1. Torus

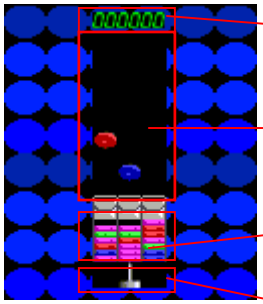
Torus의 사전적인 뜻은 원형체 이런 뜻으로 이게임에서는 타원형의 블록이 떨어집니다. 일명 도너츠 모양이죠. 색상 별로 다른 Torus가 떨어지고 슬롯에 쌓입니다. 게임의 목적은 단순히 같은 색상의 Torus를 가로줄로 만들면 없어집니다. 보통 세로줄을 잘못 인식하는 분들도 있는데 가로줄입니다. 게임의 룰이 가장 어렵다고 사람들이 그러더군요.

2. 게임 프로세서 및 UI 설명.

원제 “Torus” 여서 이것으로 했음 가칭은 “구루구루 춥”.

최고 점수, 플레이한사람 이름

난이도 설정(1.초급자, 2.중급자, 3.고급자)



현재 점수

Torus가 떨어지는 공간

Torus가 쌓이는 슬롯

Torus를 움직이는 못

마지막 점수

이름 쓰기

3. 추가 시나리오.

- 마지막 최고 점수가 되면 네트워크로 전송해서 점수 순위를 할려고 했음.
- 대전, 대결도 가능한 버전

간단하게 나마 설명 드렸습니다. 해당 시나리외에도 기타 여러 의견이 있었지만 이것 만큼은 비밀입니다. ㅎㅎㅎ 저한테도 숨기고 싶은 비밀이..

역시 우리는 프로그래머 이기 때문에 시나리오나 기획은 충분히 인지를 해야 하기 위해서는 해당 게임을 해보십시오. 그럼 간단히 게임의 룰이 들어올겁니다. 나머지 부분은 기획자들이 해야 할 몫인것죠.

본격적으로 프로그램으로 들어가기에 앞서서 “구루구루 춥”의 게임의 룰을 설명 드리겠습니다.

1. 게임의 룰

- 기본적으로 처음 시작시 슬롯의 개수는 3개, 총 7개의 Torus가 쌓입니다.
  밑의 못은 총 3개까지 찹니다.(무슨 말인지 모르시면 게임을 해보십시오)

  일정 시간이 지나면 슬롯의 개수, 쌓이는 개수 못에 차는 개수가 변경됩니다.
  순서대로 말씀드리면
  (슬롯의 개수, 슬롯에 쌓이는 개수, 못에 쌓이는 개수)

```
public final static int[][] TORUS_STAGE = {{3, 7, 3},
        {4, 8, 4}, {5, 9, 5}, {6, 10, 6}, {7, 11, 7},
        {8, 12, 8}, {9, 13, 9}, {10, 14, 10}, {11, 15, 11}, {12, 16, 12}};
```

  그리고 중간중간에 못에 쌓이는 개수가 하나 줄어듭니다. 그러니깐 처음에는
  3, 7, 3이지만 일정 시간이 지나면 3, 7, 2 그리고 일정 시간이 지나면 4, 8, 4로 변경되는 것입니다.

- 조작은 방향키 4개로서 끝납니다.
  좌, 우 : 하단의 못을 좌우로 움직입니다.
  상, 하 : 못에 있는 Torus를 위 아래로 넣다 뺐다. 합니다.

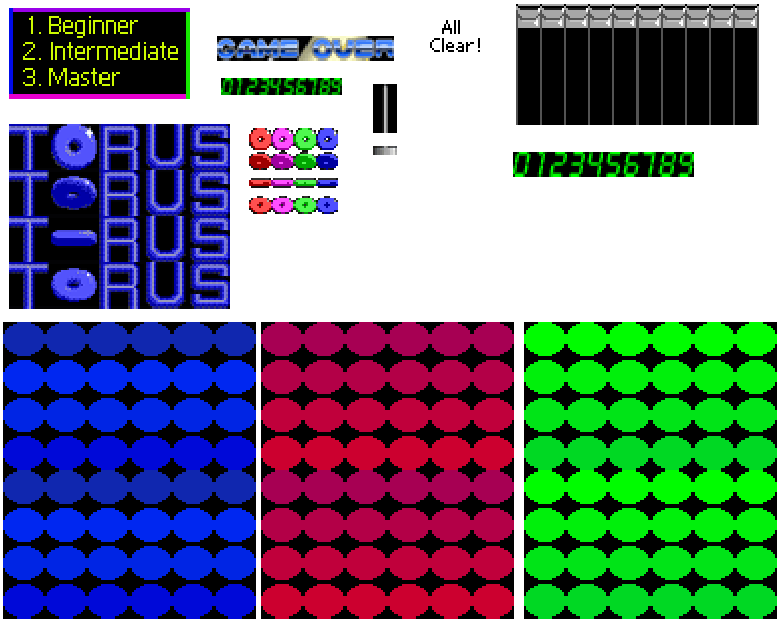
간단하죠. 이것이 룰입니다.

원래 처음 시작시에는 게임 엔진부터 만들었습니다. 왜냐하면 인트로나 게임 오버나 기타 등등은 별로 중요하지가 않죠 게임 엔진만 있면 다음 것들은 추가적으로 작업을 하면 되기 때문이죠. 그래서 강좌도 인트로부터 만들고 하는 것이 아니고 바로 엔진부터 시작해서 늘

려가는 방식으로 하겠습니다. 이것이 실전이죠. 하지만 인트로만 없을 뿐인지 모든 형식은 갖추어져 있는 것입니다. 단지 UI가 없다 뿐이죠.

2. 디자인

- 도스용 게임의 스크린샷을 해서 만들었습니다.



한 이미지에 전부 몰아 넣었습니다. 당근 파일은 전부 따로 존재합니다. 보시다시피 이미지가 많지 않은 관계로 낭비도 하지만 별 개의치 않습니다. 메모리가 남아 돌기 때문이죠. ㅎㅎㅎ 참고로 프로그램시에는 당근 짤라서 이미지를 넣었습니다. 설마 저렇게 넣었다고 생각하지 마세요..^^ 보기 좋으시라고 저렇게 포토샵 작업했습니다.

3. 엔진 프로그래밍.

- 엔진이란? 게임 프로그램에서 가장 중요한 이슈를 발생시키는 부분이죠. 엔진의 속도가 게임의 속도를 좌우합니다. 하지만 이 게임의 경우에는 워낙 간단해서..별 지장이 없다는.

그럼 원리를 생각해 보도록 합시다.

1. 위에서 떨어지는 토로스
2. 슬롯에 쌓이는 토로스
3. 못에 있는 토로스

이렇게 3가지의 토로스 위치가 있습니다.

2번은 위에 쌓이는 것과 아래 못(Screw)으로 빠져나가는 것을 생각하면 큐(Queue)가 되고, 못이 위로 올려주니 스택(Stack)이 됩니다. 또한 위에 쌓이는 것을 생까고 밑에 못과의 교

류만 생각하면 스택만 되죠. 엔진 룰이야 생각하기 나름이기 때문에 구현시에는 거의 동일하다고 생각됩니다. 메소드 이름만 다르죠.

3번은 못은 2번을 받아서 쌓다고 다시 위로 올리니 스택이 되겠죠.
2번을 큐와 스택으로 구현하고 3번을 스택으로 구현하시면 엔진은 끝이 난다는 것이죠.
의외로 엔진 알고리즘이 쉽게 끝날 것 같다는..생각이...

필요한 메소드를 보도록 합시다.

```
// 2번인 슬롯(Slot) 관련 메소드
// 여기서 전 슬롯의 큐로 하고 못쪽과 관련된 것을 스택으로 해서
// 큐용 메소드 2개 스택용 1개를 이용했지만
// 스택용 2개 그리고 쌓는 메소드를 따로 구현해도 됩니다.
public boolean slotPut(int curSlot, int torusColor)
public int slotGet(int curSlot)
public boolean slotPush(int curSlot, int torusColor)


// 3번인 나사(Screw) 관련 메소드
public void screwPush(int curSlot)
public void screwPop(int arr)
```

대충 이런 정도의 메소드가 필요하게 되죠. 간단하죠. 그럼 소스를 보도록 하겠습니다.
컴파일은 되야하니..이것저것 점 집어넣었습니다. 대충 감잡으시라고 짧게 했습니다.

TorusMIDlet.java

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

import com.xce.lcdui.*;
import com.skt.m.*;
import com.xce.io.*;

public class TorusMIDlet extends MIDlet
```

```
{

/*
** ---------------------------------------------------------------
**        변수 선언
** ---------------------------------------------------------------
*/

        Display display;                        // Display 변수
        TorusCanvas tCanvas;                    // Canvas 변수



/*
** ---------------------------------------------------------------
**   Function:
**            startApp
**
**   Description:
**            어플리케이션이 시작되면 실행됨
**
**   Input:
**
**   Return value:
**
** ---------------------------------------------------------------
*/

        public void startApp()
        {
                Device.setBacklightEnabled(false);
                BackLight.on(0);

                display = Display.getDisplay(this);
                tCanvas = new TorusCanvas(this);
                display.setCurrent(tCanvas);
```

```
        }


/*
** ---------------------------------------------------------------------
**  Function:
**          pauseApp
**
**  Description:
**          어플리케이션이 멈출경우 호출됨(잘 동작하지 않음)
**
**  Input:
**
**  Return value:
**
** ---------------------------------------------------------------------
*/
        public void pauseApp() {}

/*
** ---------------------------------------------------------------------
**  Function:
**          resumeApp
**
**  Description:
**          어플리케이션 pauseApp() 후에 복귀 할때 호출됨(실제로 동작하지 않음)
**
**  Input:
**
**  Return value:
**
** ---------------------------------------------------------------------
*/

        public void resumeApp(){}
```

```
/*
** ----------------------------------------------------------------
**   Function:
**            destroyApp
**
**   Description:
**            어플리케이션 종료 될때 호출됨
**
**   Input:
**
**   Return value:
**
** ----------------------------------------------------------------
*/
          public void destroyApp(boolean con){}

}
```

TorusCanvas.java

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

import com.xce.lcdui.*;
import com.xce.io.*;
import com.skt.m.*;

public class TorusCanvas extends Canvas
{

/*
** ----------------------------------------------------------------
**        define 선언
** ----------------------------------------------------------------
*/
```

```
        // 상태
        public final static int TORUS_INTRO   = 0;              // 인트로
        public final static int TORUS_READY   = 1;              // 시작
        public final static int TORUS_GO      = 2;              // 게임 중.
        public final static int TORUS_OVER    = 3;              // 게임 오버


        // Bar : 가로, 세로   Screw : 세로.
        public final static int[][] TORUS_STAGE = {{3, 7, 3},
                {4, 8, 4}, {5, 9, 5}, {6, 10, 6}, {7, 11, 7},
                {8, 12, 8}, {9, 13, 9}, {10, 14, 10}, {11, 15, 11}, {12, 16, 12}};


        public final static int MAX_COLOR              = 4;     // Torus 최대 색상 개수.
        public final static int LEVEL_UP_CONDITION     = 300;   // 레벨 업 조건.

/*
** ----------------------------------------------------------------
**      Class 선언
** ----------------------------------------------------------------
*/

        TorusMIDlet       tMIDlet;
        TorusEngine       tEngine;


/*
** ----------------------------------------------------------------
**      변수 선언
** ----------------------------------------------------------------
*/

        int statusTorus = TORUS_INTRO;


/*
```

```
** ----------------------------------------------------------------------
**        Image 선언
** ----------------------------------------------------------------------
*/



/*
** ----------------------------------------------------------------------
**        Method 선언
** ----------------------------------------------------------------------
*/



/*
** ----------------------------------------------------------------------
**   Function:
**          TorusCanvas
**
**   Description:
**              생성자.
**
**   Input:
**
**   Return value:
**
** ----------------------------------------------------------------------
*/

          TorusCanvas(TorusMIDlet tMIDlet)
          {
                    this.tMIDlet = tMIDlet;
                    tEngine = new TorusEngine(this);
          }
```

```
/*
** ---------------------------------------------------------------
**  Function:
**              paint
**
**  Description:
**
**  Input:
**
**  Return value:
**
** ---------------------------------------------------------------
*/
        public void paint(Graphics g)
        {
                switch(statusTorus)
                {
                        case TORUS_INTRO:
                                break;

                        case TORUS_READY:
                                break;

                        case TORUS_GO:
                                break;

                        case TORUS_OVER:
                                break;
                }
        }


/*
** ---------------------------------------------------------------
**  Function:
```

```
**             keyPressed
**
**  Description:
**             -  키 누름 처리
**
**  Input:
**                 // SK-VM Specific
**                 KEY_CLR          = 8;
**
**                 KEY_POUND        = 35;
**                 KEY_NUM0         = 48;
**                 KEY_NUM1         = 49;
**                 KEY_NUM2         = 50;
**                 KEY_NUM3         = 51;
**                 KEY_NUM4         = 52;
**                 KEY_NUM5         = 53;
**                 KEY_NUM6         = 54;
**                 KEY_NUM7         = 55;
**                 KEY_NUM8         = 56;
**                 KEY_NUM9         = 57;
**                 KEY_STAR         = 42;
**
**                 KEY_COML         = 129;
**                 KEY_COMC         = 130;    // RESERVED
**                 KEY_COMR         = 131;
**
**                 KEY_UP           = 141;
**                 KEY_LEFT         = 142;
**                 KEY_RIGHT        = 145;
**                 KEY_DOWN         = 146;
**                 KEY_FIRE         = 148;
**
**                 KEY_CALL         = 190;
**                 KEY_END          = 191;
**
**                 KEY_FLIP_OPEN    = 192;
```

```
**                KEY_FLIP_CLOSE  = 193;
**                KEY_VOL_UP      = 194;
**                KEY_VOL_DOWN    = 195;
**
**   Return value:
**
** ----------------------------------------------------------------------
*/



        public void keyPressed(int keyCode)
        {
                switch(statusTorus)
                {
                        case TORUS_INTRO:
                                break;

                        case TORUS_GO:
                                tEngine.keyPressed(keyCode);
                                break;

                        case TORUS_OVER:
                                break;
                }
        }

/*
** ----------------------------------------------------------------------
**   Function:
**           keyReleased
**
**   Description:
**
**
**   Input:
**
```

```
**   Return value:
**
** ----------------------------------------------------------------
*/

        public void keyReleased(int keyCode)
        {
                switch(statusTorus)
                {
                        case TORUS_INTRO:
                                break;

                        case TORUS_GO:
                                tEngine.keyReleased(keyCode);
                                break;

                        case TORUS_OVER:
                                break;
                }
        }
}
```

TorusEngine.java

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

import com.xce.lcdui.*;
import com.xce.io.*;
import com.skt.m.*;

public class TorusEngine
{

/*
** ----------------------------------------------------------------
```

```
**      define 선언
** --------------------------------------------------------------------
*/

/*
** --------------------------------------------------------------------
**      Class 선언
** --------------------------------------------------------------------
*/

        TorusCanvas       tCanvas;



/*
** --------------------------------------------------------------------
**      변수 선언
** --------------------------------------------------------------------
*/

        // SLOT 관련.
        // 미리 배열을 최대 값으로 할당하여 속도를 빠르게 한다.
        // 단 이럴경우 해당 값들의 최대 최소 현재 위치등을 확실히 관리해야 한다.
        // 여기서 라인은 높이..즉 Torus가 쌓이는 개수를 의미한다.
        int[][] TORUS_SLOT = new int[12][16];       // SLOT의 2차원 배열.
        int[] slotCurLine = new int[16];       // 현재 각각의 SLOT에 쌓여 있는 최대 라인.
        int[] slotDelLine = new int[16];       // 지워져야할 라인을 기억한다.
        int slotMaxLine;                       // SLOT의 가로 라인의 최대  개수.
        int slotCount;                                                  // SLOT의 개수

        // Screw 관련.
        int[] TORUS_SCREW = new int[12];  // 나사.(최대 개수)

        int screwCurStatck;                    // 스택의 현재 위치
        int maxOfscrewStack;                   // Screw에 들어갈 Torus 최대 개수.
        int screwPos;                          // 나사 위치
```

```
/*
** ---------------------------------------------------------------------
**  Function:
**          TorusEngine
**
**  Description:
**
**
**  Input:
**
**  Return value:
**
** ---------------------------------------------------------------------
*/

        TorusEngine(TorusCanvas tCanvas)
        {
                this.tCanvas = tCanvas;
        }

/*
** ---------------------------------------------------------------------
**  Function:
**              - slotPut
**
**  Description:
**              - Slot의 큐의 Put
**
**  Input:
**              - curSlot : 현재 선택된 슬롯.
**              - torusColor : 현재 슬롯에 들어갈 토로스 색상.
**
**  Return value:
**              - 성공, 실패.
**
** ---------------------------------------------------------------------
```

```
*/

	public boolean slotPut(int curSlot, int torusColor)
	{
		// 슬롯이 꽉찬경우, 현재 슬롯 개수보다 많을경우, 최대 색상을 넘어갔을 경우
//		if(slotCurLine[curSlot] >= slotMaxLine || curSlot >= slotCurLine.length ||
//			torusColor > TorusCanvas.MAX_COLOR)
//			return false;

		// 실제적으로는 아래 에러만 검출하면 됨.
		if(slotCurLine[curSlot] >= slotMaxLine)
			return false;

		// SLOT에 쌓는다.
		TORUS_SLOT[curSlot][slotCurLine[curSlot]++] = torusColor;

		return true;
	}


/*
** ---------------------------------------------------------------------
**  Function:
**		- slotGet
**
**  Description:
**		- Slot의 큐의 Get
**
**  Input:
**		- curSlot : 현재 선택된 슬롯.
**
**  Return value:
**		- 0 : 이상..발견..
**		- 1 Over : Color Value
**
** ---------------------------------------------------------------------
```

```
*/

	public int slotGet(int curSlot)
	{
		// 슬롯이 빈경우.
//		if(TORUS_SLOT[curSlot][0] == 0 || curSlot >= slotCurLine.length)
//			return 0;

		// 실제적으로는 아래 에러만 검출하면 됨.
		if(TORUS_SLOT[curSlot][0] == 0)
			return 0;

		// 맨 밑에 있는 Torus의 칼라값을 얻는다.
		int colorValue = TORUS_SLOT[curSlot][0];

		// 한칸씩 내려준다.
		System.arraycopy(TORUS_SLOT[curSlot], 1,
		    TORUS_SLOT[curSlot], 0, slotCurLine[curSlot]--);

		return colorValue;
	}



/*
** ------------------------------------------------------------------
**  Function:
**		- slotPush
**
**  Description:
**		- Slot의 스택의 Push
**
**  Input:
**		- curSlot : 현재 선택된 슬롯.
**		- torusColor : 현재 슬롯에 들어갈 토로스 색상.
**
```

```
**  Return value:
**                  - 성공, 실패.
**
** ----------------------------------------------------------------
*/
        public boolean slotPush(int curSlot, int torusColor)
        {
                // 슬롯이 꽉찬경우, 현재 슬롯 개수보다 많을경우, 최대 색상을 넘어갔을 경우
//              if(slotCurLine[curSlot] >= slotMaxLine ||curSlot >= slotCurLine.length ||
//                      torusColor > TorusCanvas.MAX_COLOR)
//                      return false;

                // 실제적으로는 아래 에러만 검출하면 됨.
                if(slotCurLine[curSlot] >= slotMaxLine)
                        return false;


                for(int i = slotCurLine[curSlot]++; i >= 1; i--)
                        TORUS_SLOT[curSlot][i] = TORUS_SLOT[curSlot][i -1];

                TORUS_SLOT[curSlot][0] = torusColor;

                return true;
        }




/*
** ----------------------------------------------------------------
**  Function:
**                  - screwPush
**
**  Description:
**                  - Screw 스택의 Push
**
```

```
**  Input:
**                  - curSlot : 현재 선택된 슬롯.
**
**  Return value:
**
** ----------------------------------------------------------------
*/

        public void screwPush(int curSlot)
        {
                if(screwCurStatck >= maxOfscrewStack)
                        return;

                // SLOT으로 부터 Get한다.
                int popColor = slotGet(curSlot);

                if(popColor > 0)
                        TORUS_SCREW[screwCurStatck++] = popColor;
        }

/*
** ----------------------------------------------------------------
**  Function:
**                  - screwPop
**
**  Description:
**                  - Screw 스택의 Pop
**
**  Input:
**                  - curSlot : 현재 선택된 슬롯.
**
**  Return value:
**
** ----------------------------------------------------------------
*/
        public void screwPop(int curSlot)
```

```
		{
			if(screwCurStatck <= 0)
				return;

			// SLOT으로 Push한다.
			if(slotPush(curSlot, TORUS_SCREW[screwCurStatck - 1]))
				TORUS_SCREW[--screwCurStatck] = 0;
		}



/*
** ---------------------------------------------------------------
**  Function:
**		paint
**
**  Description:
**
**
**  Input:
**
**  Return value:
**
** ---------------------------------------------------------------
*/

		public void paint(Graphics g)
		{
		}

/*
** ---------------------------------------------------------------
**  Function:
**		keyPressed
**
**  Description:
```

```
**
**
**   Input:
**
**   Return value:
**
** ----------------------------------------------------------------
*/
        public void keyPressed(int keyCode)
        {
                switch(keyCode)
                {
                        // Screw에 있는 것을 SLOT으로 올린다.
                        case Canvas.KEY_UP://     = 141;
                                screwPop(screwPos);
                                break;

                        // Screw를 좌로 움직인다.
                        // 맨 왼쪽으로 갈 경우에는 더 이상 가지 못한다.
                        case Canvas.KEY_LEFT://   = 142;
                                screwPos = (--screwPos < 0)?0:screwPos;
                                break;

                        // Screw를 우로 움직인다.
                        // 맨 오른쪽으로 갈 경우에는 더 이상 가지 못한다.
                        case Canvas.KEY_RIGHT:// = 145;
                                screwPos = (++screwPos >= slotCount)?
                                        slotCount - 1:screwPos;
                                break;

                        // SLOT에 있는 것을 Screw로 내린다.
                        case Canvas.KEY_DOWN:// = 146;
                                screwPush(screwPos);
                                break;
                }
        }
```

```
/*
** ---------------------------------------------------------------
**   Function:
**              keyReleased
**
**   Description:
**
**
**   Input:
**
**   Return value:
**
** ---------------------------------------------------------------
*/
          public void keyReleased(int keyCode)
          {
          }
}
```

대충 쭉 보니..정리가 잘되었는지 모르겠네요..이거 쓸려고 기존 소스를 정리하면서 하니 강좌를 쓰는 속도도 느려지고 이래저래 귀찮지만 소스가 더욱더 좋아지는 것 같아서 좋네요.

그때는 한다고 했는데..미흡한 부분은 언제나 나중에 보이기 마련인 것 같아여 마음의 여유를 두고 프로그래밍을 해야지 볼게 보인다는..역시 나무만 보니 가끔씩 숲도 봐줘야 한다는 생각이 듭니다. 하나만 보면 손해죠..^^

다음 강좌는 해보면서 조금씩 나아가 보도록 하죠..원래 이렇게 느리게 진행되면
강좌를 보시는 분들이 답답해 하기는 하는데 저 또한 준비하는게 시간의 제약이 있어서
한꺼번에 많은 양을 제공해 드릴수가 없네요..^^

그럼 이만..즐건 하루 되십시오..핑크레드.

닉네임 : 핑크레드
성격 : 왕소심, 왕삐짐, 이중인격, 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.