1. 유니티

(1) UnityPlayerActivity.java

```
package com.kimsunghoon.helloworldwearable;

import com.unity3d.player.*;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.res.Configuration;
import android.database.Cursor;
import android.graphics.PixelFormat;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.net.Uri;
import android.os.Bundle;
import android.os.Vibrator;
import android.provider.MediaStore;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.Window;
import android.view.WindowManager;

import java.util.Timer;
import java.util.TimerTask;

public class UnityPlayerActivity extends Activity
{
          protected UnityPlayer mUnityPlayer; // don't change the name of this variable; referenced from
native code

    //센서 매니저를 위한 변수들
    SensorManager mSensorManager;
    Sensor mSensorGyro;
    Sensor mSensorAccelerometer;
    Sensor mSensorOrientation;
    Sensor mSensorHeartrate;

    //로그를 위한 스트링 매크로
    private static final String TAG="CELLPHONE_TEST";

    //0.1초마다 전송해주는 타이머
    private TimerTask mTask;
    private Timer mTimer;

    //현재 움직여야 하는 object의 이름. 내 것인지 아닌지 판별하는 용도
    String currentObjectName ="";
```

```
      // Setup activity layout      @Override
       protected void onCreate (Bundle savedInstanceState) {
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    super.onCreate(savedInstanceState);
     getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    getWindow().setFormat(PixelFormat.RGBX_8888); // <--- This makes xperia play happy

    mUnityPlayer = new UnityPlayer(this);
    if (mUnityPlayer.getSettings().getBoolean("hide_status_bar", true)) {
        setTheme(android.R.style.Theme_NoTitleBar_Fullscreen);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
    }

    setContentView(mUnityPlayer);
    mUnityPlayer.requestFocus();

    // 센서 매니저와 센서들을 등록 (자이로스코프, 방향, 심장 박동)
    mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    mSensorAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
    mSensorOrientation = mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);
    mSensorHeartrate = mSensorManager.getDefaultSensor(Sensor.TYPE_HEART_RATE);
}

      // Quit Unity
      @Override protected void onDestroy ()
      {
                mUnityPlayer.quit();
                super.onDestroy();
      }

      // Pause Unity
      @Override protected void onPause()
      {
                super.onPause();
    mTimer.cancel();
    mSensorManager.unregisterListener(mSensorListener);
                mUnityPlayer.pause();
      }

private double x,y,z;
float previousJumpCounter=0;
float jumpCounter=0;
float heartrate, previousHeartrate;
long currentTime, lastTime;

      // Resume Unity
      @Override protected void onResume()
      {
                super.onResume();
    //센서매니저에 방향과 가속도계 센서 리스너를 등록
```

```
        mSensorManager.registerListener(mSensorListener,                                    mSensorOrientation,
SensorManager.SENSOR_DELAY_NORMAL);
        mSensorManager.registerListener(mSensorListener,                                  mSensorAccelerometer,
SensorManager.SENSOR_DELAY_GAME);
        mTask = new TimerTask() {
            @Override
            public void run() {
                if(currentObjectName.equals("")==false) {
                    //가속도계 데이터를 불러와서 object의 position을 바꿔주는 역할. 0.1초마다 수행됨
                    x=accel_data[0];
                    y=accel_data[2];
                    z=30;

                    final String leafPositionMessage = Double.toString(x) + "," + Double.toString(y) + ","
+ Double.toString(z);
                    UnityPlayer.UnitySendMessage(currentObjectName,                    "ChangeLeafPosition",
leafPositionMessage);

                    final String cameraRotationMessage = Float.toString(sendingResult[0]);
                    UnityPlayer.UnitySendMessage(currentObjectName,                         "CameraRotation",
cameraRotationMessage);

                    //@deprecated. 가속도계를 이용한 object를 흔들게 하는 효과. 서버 측으로 기능이 이전됨
                    /*
                    currentTime = System.currentTimeMillis();
                    jumpCounter = sendingResult[3];
                    if (previousJumpCounter != jumpCounter && currentTime - lastTime > 1000
                            && jumpCounter > 6 && jumpCounter < 10) {
                        UnityPlayer.UnitySendMessage(currentObjectName, "AccelCamera", "");
                        lastTime = System.currentTimeMillis();
                        previousJumpCounter = jumpCounter;
                    }
                    */

                    //@deprecated. heartrate를 이용한 배경 깜빡이기
                    /*
                    heartrate = sendingResult[4];
                    if (heartrate == 0) {
                        UnityPlayer.UnitySendMessage(currentObjectName, "CancelHeartrate", "");
                    } else if (heartrate != 0 && previousHeartrate != heartrate) {
                        previousHeartrate = heartrate;
                        UnityPlayer.UnitySendMessage(currentObjectName, "SetHeartrate", Float.toString(60
/ heartrate));
                    }
                    */
                }
            }
        };
        mTimer=new Timer();
        //mTask는 0.1초마다 수행됨
        mTimer.schedule(mTask, 0, 100);
```

```
                mUnityPlayer.resume();
        }

        // This ensures the layout will be correct.
        @Override public void onConfigurationChanged(Configuration newConfig)
        {
                super.onConfigurationChanged(newConfig);
                mUnityPlayer.configurationChanged(newConfig);
        }

        // Notify Unity of the focus change.
        @Override public void onWindowFocusChanged(boolean hasFocus)
        {
                super.onWindowFocusChanged(hasFocus);
                mUnityPlayer.windowFocusChanged(hasFocus);
        }

        // For some reason the multiple keyevent type is not supported by the ndk.
        // Force event injection by overriding dispatchKeyEvent().
        @Override public boolean dispatchKeyEvent(KeyEvent event)
        {
                if (event.getAction() == KeyEvent.ACTION_MULTIPLE)
                        return mUnityPlayer.injectEvent(event);
                return super.dispatchKeyEvent(event);
        }

    //현재 내 object의 이름을 등록. 내 object만 움직여야 하기 때문에 사용
    public void setCurrentObjectName(String name)
    {
        this.currentObjectName=name;
        Log.d(TAG, "setCurrentObjectName : " + name);
    }

         // Pass any events not handled by (unfocused) views straight to UnityPlayer
         @Override public boolean onKeyUp(int keyCode, KeyEvent event)             { return
mUnityPlayer.injectEvent(event); }
         @Override public boolean onKeyDown(int keyCode, KeyEvent event)         { return
mUnityPlayer.injectEvent(event); }
         @Override public boolean onTouchEvent(MotionEvent event)                       { return
mUnityPlayer.injectEvent(event); }
         /*API12*/    public    boolean    onGenericMotionEvent(MotionEvent    event)        {    return
mUnityPlayer.injectEvent(event); }


    float[] accel_data=new float[3];
    float[] temp_orientation=new float[3];
    float inputHeartrate;
    float[] sendingResult=new float[5];

    SensorEventListener mSensorListener=new SensorEventListener() {
        @Override
```

```
    public void onSensorChanged(SensorEvent event) {
        //@Deprecated
        if(event.sensor.getType()==Sensor.TYPE_ORIENTATION) {
            temp_orientation=event.values;
        }
        //가속도계가 들어오면 accel_data에 저장한 후, mTask에서 참조하도록 함
        else if(event.sensor.getType()==Sensor.TYPE_LINEAR_ACCELERATION)
        {
            accel_data=event.values.clone();
        }
        //@Deprecated
        else if(event.sensor.getType()==Sensor.TYPE_HEART_RATE) {
            inputHeartrate=event.values[0];
        }

        sendingResult[0]=temp_orientation[0];
        sendingResult[1]=temp_orientation[1];
        sendingResult[2]=temp_orientation[2];
        sendingResult[3]=accel_data[2];
        sendingResult[4]=inputHeartrate;
    }
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }
};

private static final int CUSTOM_TEXTURE = 1;
private static final int CUSTOM_BACKGROUND = 2;

private Uri mImageCaptureUri;
private String openGallerySendToUnityObject;

//휴대폰 진동 울리게 하는 함수
public void vibratePhone(String argv) {
    Vibrator vibe = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
    vibe.vibrate(500);
}

//object의 텍스쳐를 갤러리에서 불러와 지정하는 함수.
public void setCustomTexture(String toObjectName) {
    this.openGallerySendToUnityObject=toObjectName;
    Intent intent=new Intent(Intent.ACTION_PICK);
    intent.setType(MediaStore.Images.Media.CONTENT_TYPE);
    startActivityForResult(intent, CUSTOM_TEXTURE);
}

//@Deprecated. object의 배경을 갤러리에서 불러와 지정하는 함수.
public void setCustomBackground(String toObjectName) {
    this.openGallerySendToUnityObject=toObjectName;
    Intent intent=new Intent(Intent.ACTION_PICK);
    intent.setType(MediaStore.Images.Media.CONTENT_TYPE);
```

```
        startActivityForResult(intent, CUSTOM_BACKGROUND);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if(resultCode != RESULT_OK)
        {
            Log.d(TAG, "resultCode is invalid. Errcode: " + resultCode);
            return;
        }

        //갤러리로부터 선택한 이미지의 절대 경로를 받아옴
        mImageCaptureUri = data.getData();
        Log.d(TAG, mImageCaptureUri.toString());
        Log.d(TAG, "URI!!!! : " + mImageCaptureUri.toString());

        //절대경로를 받아옴
        String path=getPathFromUri(mImageCaptureUri);
        Log.d(TAG, "절대경로 : " + path);
        UploadFile uploadFile=new UploadFile(openGallerySendToUnityObject);

        Log.d(TAG, "switch request code");
        //call 된 함수에 따라 uploadFile 클래스를 이용해 백그라운드에서 올려줌
        switch(requestCode) {
            case CUSTOM_TEXTURE:
                uploadFile.execute(path, "SetCustomTextureFromAndroid");
                break;
            case CUSTOM_BACKGROUND:
                uploadFile.execute(path, "SetCustomBackgroundFromAndroid");
                break;
            default:
                Log.d(TAG, "default");
                break;
        }
    }

    //Uri를 절대경로로 바꿔주는 함수
    public String getPathFromUri(Uri uri) {
        Cursor cursor = getContentResolver().query(uri, null, null, null, null);
        cursor.moveToNext();
        String path = cursor.getString(cursor.getColumnIndex("_data"));
        cursor.close();
        return path;
    }
}
```

(2) UploadFile.java

```
package com.kimsunghoon.helloworldwearable;

import android.os.AsyncTask;
```

```
import android.util.Log;

import com.unity3d.player.UnityPlayer;

import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

/**
 * Created by KimSunghoon on 2015. 7. 26..
 */
public class UploadFile extends AsyncTask<String, Void, String> {
    String fileName;
    HttpURLConnection conn = null;
    DataOutputStream dos = null;
    String lineEnd = "₩r₩n";
    String twoHyphens = "--";
    String boundary = "*****";
    int bytesRead, bytesAvailable, bufferSize;
    byte[] buffer;
    int maxBufferSize = 1 * 1024 * 1024;
    String uploadedFileName;

    private final static String serverUrl="http://52.69.154.248/";
    private final static String uploadServerUri = serverUrl+"uploadtoserver.php";//서버컴퓨터의 ip주소
    private final static String TAG="UploadFile";
    int serverResponseCode=0;
    String callUnityMethod;
    String openGallerySendToUnityObject;

    UploadFile(String galleryObject) {
        this.openGallerySendToUnityObject=galleryObject;
    }

    @Override
    protected String doInBackground(String... sourceFileUri) {
        fileName = sourceFileUri[0];
        callUnityMethod=sourceFileUri[1];
        File sourceFile = new File(sourceFileUri[0]);
        Log.d(TAG, "소스파일 이름: " + sourceFile.getName());
        uploadedFileName=sourceFile.getName();
        if (!sourceFile.isFile()) {
            Log.d(TAG, "source file is invalid.");
            return "";
        } else {
            try {
                // open a URL connection to the Servlet
```

```
FileInputStream fileInputStream = new FileInputStream(sourceFile);
URL url = new URL(uploadServerUri);

// Open a HTTP  connection to  the URL
conn = (HttpURLConnection) url.openConnection();
conn.setDoInput(true); // Allow Inputs
conn.setDoOutput(true); // Allow Outputs
conn.setUseCaches(false); // Don't use a Cached Copy
conn.setRequestMethod("POST");
conn.setRequestProperty("Connection", "Keep-Alive");
conn.setRequestProperty("ENCTYPE", "multipart/form-data");
conn.setRequestProperty("Accept-charset", "UTF-8");
conn.setRequestProperty("Content-Type", "multipart/form-data;boundary=" + boundary);
conn.setRequestProperty("uploaded_file", fileName);

dos = new DataOutputStream(conn.getOutputStream());

dos.writeBytes(twoHyphens + boundary + lineEnd);
dos.writeBytes("Content-Disposition: form-data; name=₩"uploaded_file₩";filename=₩""
        + new String(fileName.getBytes("UTF-8"),"ISO-8859-1") + "₩"" + lineEnd);
dos.writeBytes(lineEnd);

// create a buffer of  maximum size
bytesAvailable = fileInputStream.available();

bufferSize = Math.min(bytesAvailable, maxBufferSize);
buffer = new byte[bufferSize];

// read file and write it into form...
bytesRead = fileInputStream.read(buffer, 0, bufferSize);

while (bytesRead > 0) {
    dos.write(buffer, 0, bufferSize);
    bytesAvailable = fileInputStream.available();
    bufferSize = Math.min(bytesAvailable, maxBufferSize);
    bytesRead = fileInputStream.read(buffer, 0, bufferSize);
}

// send multipart form data necessary after file data...
dos.writeBytes(lineEnd);
dos.writeBytes(twoHyphens + boundary + twoHyphens + lineEnd);

//close the streams //
dos.flush();
dos.close();
fileInputStream.close();

// Responses from the server (code and message)
serverResponseCode = conn.getResponseCode();
String serverResponseMessage = conn.getResponseMessage();
```

```
                Log.i("uploadFile", "HTTP Response is : "
                        + serverResponseMessage + ": " + serverResponseCode);

                if (serverResponseCode == 200) {
                    Log.d(TAG, "file upload completed.");
                }

                // get response
                int ch;
                InputStream is = conn.getInputStream();
                StringBuffer b =new StringBuffer();
                while( ( ch = is.read() ) != -1 ){
                    b.append( (char)ch );
                }
                String s=b.toString();
                Log.e(TAG, "result = " + s);

            } catch (MalformedURLException ex) {
                Log.e("Upload file to server", "error: " + ex.getMessage(), ex);
            } catch (Exception e) {
                Log.e("Upload file to server Exception", "Exception : "
                        + e.getMessage(), e);
            }
        } // End else block
        return callUnityMethod;
    }

    @Override
    protected void onPostExecute(String callUnityMethod) {
        super.onPostExecute(callUnityMethod);
        String uploadedPath=serverUrl+"uploads/"+uploadedFileName;
        UnityPlayer.UnitySendMessage(openGallerySendToUnityObject, callUnityMethod, uploadedPath);
        uploadedFileName="";
    }
}
```

2. 유니티

(1) AndroidPluginManager.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class AndroidPluginManager : MonoBehaviour
{
#if UNITY_ANDROID
          private AndroidJavaObject curActivity;
          private AndroidJavaClass firstPluginJc;
          private PhotonView photonView;
```

```
	Vector3 transform_vector;
	public bool isChangeLeafPositionEnabled=true;
	Vector3 scaleVector;
	public float y_rotation;
	public bool isBeingMovedByServer=false;

	const float ORIGINAL_SIZE = 10.0f;
	const float ALPHA = 2.0f;
	const float DIVIDEBY = 30.0f;
	void Start()
	{
		photonView = gameObject.GetComponent<PhotonView> ();
		AndroidJavaClass jc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
		curActivity = jc.GetStatic<AndroidJavaObject>("currentActivity");
		transform_vector = new Vector3 (0, 0, 0);
		scaleVector=new Vector3();
		Debug.Log ("AndroidPluginManager this called!!!!!");
		if (photonView.isMine == true) {
			SetCurrentObjectNameToAndroid (gameObject.name);
		}
	}

	//@Deprecated.
	public void AccelCamera(string parameter)
	{
//		main_camera_script.CameraShakeEffect();
	}

	//@Deprecated.
	public void CameraRotation(string parameter)
	{
//		if (Network.isServer) {
//			var value_y = System.Convert.ToSingle (parameter);
//			if(value_y!=0)
//			{
//				y_rotation=value_y;
//			}
//			main_camera_script.CameraRotation (value_y);
//		}
	}

	//Send my object name to android phone.
	public void SetCurrentObjectNameToAndroid(string name) {
		curActivity.Call ("setCurrentObjectName", name);
	}

	//Change game object's position.
	public void ChangeLeafPosition(string parameter){
		if (photonView.isMine==true && isChangeLeafPositionEnabled == true) {
			string[] vector_array = parameter.Split (',');
			var value_x = System.Convert.ToSingle (vector_array [0]);
```

```
                        var value_y = System.Convert.ToSingle (vector_array [1]);
                        var value_z=0;

                        transform_vector.Set (value_x*100.0f, value_y*100.0f, 0.0f);

                        //Control object's force

                        //가속도계의 절대값이 2보다 작거나 같으면 물체는 정지함
                        if(Mathf.Abs(value_x)<=2 && Mathf.Abs (value_y)<=2)
                        {
                                gameObject.GetComponent<Rigidbody>().velocity=new    Vector3(0.0f,
0.0f, 0.0f);
                        }
                        //아닐 경우에는 100배의 힘을 작용시켜서 움직임
                        else{
                        gameObject.GetComponent<Rigidbody>().AddForce(transform_vector,
ForceMode.Impulse);
                        }

                        if(isBeingMovedByServer==false) {
                                //Control object's scale
                                scaleVector.Set(ORIGINAL_SIZE+ALPHA*Mathf.Abs(value_x),
                                        ORIGINAL_SIZE+ALPHA*Mathf.Abs(value_y),
                                        ORIGINAL_SIZE);
                                gameObject.transform.localScale=scaleVector;
                        }
                }
                else {
                        return;
                }
        }

        //When leaf is stopped, stay 2 seconds then start again.
        public IEnumerator StopLeafCallByLeafControl() {
                isChangeLeafPositionEnabled = false;
                yield return new WaitForSeconds (2);
                isChangeLeafPositionEnabled = true;
        }

        //@depreacted.
        public void SetHeartrate(string parameter){
//                  var heartrate = System.Convert.ToSingle (parameter);
//                  wall_script.SetHeartrate (heartrate);
        }

        //@deprecated.
        public void CancelHeartrate(string parameter) {
//                  wall_script.CancelHeartrate ();
        }

        //Vibrate my phone.
```

```
	public void VibratePhone()
	{
		curActivity.Call ("vibratePhone", "");
	}
#endif
}
```

(2) ChangeBackground.cs

```
using UnityEngine;
using System.Collections;

public class ChangeBackground : MonoBehaviour {
	private PhotonView pv;
	GameObject alphaWall;
	Room curRoom;
	// Use this for initialization
	void Start () {
		alphaWall = GameObject.Find ("AlphaWall") as GameObject;
		pv = GetComponent<PhotonView> ();
		curRoom = PhotonNetwork.room;
		bool isCustomBack = System.Convert.ToBoolean (curRoom.customProperties
["ISCUSTOMBACK"].ToString());
		string backgroundOfNewUser = curRoom.customProperties ["BACKGROUND"].ToString();
		pv.RPC ("SetBackground", PhotonTargets.All, backgroundOfNewUser, isCustomBack);
	}

	//Set background. Download texture from Amazon server.
	[PunRPC]
	public void SetBackground(string background, bool isCustomBack) {
		if (isCustomBack == true) {
			StartCoroutine(DownloadBackground(background));
		} else {
			Texture texture=Resources.Load (background) as Texture;
			alphaWall.GetComponent<Renderer>().material.mainTexture=texture;
		}
	}

	IEnumerator DownloadBackground(string url) {
		string escapeUriString = System.Uri.EscapeUriString (url);
		WWW www = new WWW (escapeUriString);

		// Wait for download to complete
		yield return www;

		// assign texture
		alphaWall.GetComponent<Renderer>().material.mainTexture = www.texture;
	}
}
```

(3) FauxGravityAttractor.cs

```
using UnityEngine;
using System.Collections;

public class FauxGravityAttractor : MonoBehaviour {

	public float gravity;

	public void Attract(Transform body) {

		Vector3 gravityUp = (body.position - transform.position).normalized;

		Vector3 localUp = body.up;

		body.GetComponent<Rigidbody>().AddForce(gravityUp * gravity);

		Quaternion targetRotation = Quaternion.FromToRotation(localUp,gravityUp) *
body.rotation;
		body.rotation = Quaternion.Slerp(body.rotation,targetRotation,50f * Time.deltaTime );
	}
}
```

(4) FauxGravityBody.cs

```
using UnityEngine;
using System.Collections;

[RequireComponent (typeof (Rigidbody))]
public class FauxGravityBody : MonoBehaviour {

	public FauxGravityAttractor attractor;
	private Transform myTransform;

	void Start () {
		GetComponent<Rigidbody>().useGravity = false;

		myTransform = transform;
//		attractor = GameObject.Find("Planet").GetComponent<FauxGravityAttractor>();
	}

	void FixedUpdate () {
		if (attractor){
			attractor.Attract(myTransform);
		}
	}
}
```

(5) GameMgr.cs

```
using System.Collections.Generic;
using UnityEngine;
using System.Collections;
```

```
public class GameMgr : MonoBehaviour {

	const string PLAYERCUBE="PlayerCube";
	const string PLAYERSPHERE="LeavesA";
	const int RIGHT_CLICK = 1;
	private bool isServer;

	public AudioClip[] audioclip;
	int currentSongNumber=0;
	const int numberOfTotalSong=3;
	AudioSource audioSource;
	//For LeafControl
	public Dictionary<int, TransformInfo> userTransformInfo;

	void Start() {
		//Reconnect network.
		PhotonNetwork.isMessageQueueRunning = true;

		//isServer? create server object : create client object.
		isServer = (bool)PhotonNetwork.player.customProperties ["ISSERVER"];
		if (isServer == true) {
			Debug.Log ("I am SERVER");
			PhotonNetwork.Instantiate ("ServerObject",
						new Vector3(0.0f, 0.0f, 0.0f),
						Quaternion.identity,
						0);
		} else {
			StartCoroutine (this.CreateLeaf ());
		}

		//For playing background music
		audioSource = gameObject.GetComponent<AudioSource> ();

		//To save each object's position.
		//If it's not my object, refer to userTransformInfo[] and change each position.
		userTransformInfo=new Dictionary<int, TransformInfo>();

		PhotonPlayer[] players = PhotonNetwork.playerList;
		foreach (PhotonPlayer player in players) {
			TransformInfo transformInfo = new TransformInfo ();
			if(userTransformInfo.ContainsKey(player.ID)==false)
				userTransformInfo.Add (player.ID, transformInfo);
		}
	}

	void Update() {
		if(isServer==true)
		{
			//Escape the room.
			if (Input.GetKey(KeyCode.Escape)) {
				PhotonNetwork.LeaveRoom();
```

```
                    StartCoroutine(LoadLobbyForServer());
                }

                //Mouse rightclick, play background music.
                //If playing, then stop the music.
                if (Input.GetMouseButtonUp(RIGHT_CLICK)) {
                    if(audioSource.isPlaying==false) {
                        AudioPlay(currentSongNumber%numberOfTotalSong);
                        currentSongNumber++;
                    }
                    else {
                        audioSource.Stop();
                    }
                }
            }
        }

        //Play background music when mouse right button clicks.
        void AudioPlay(int number) {
            audioSource.clip = audioclip [number];
            audioSource.Play ();
        }

        //Exit the room, then load lobby for server.
        IEnumerator LoadLobbyForServer() {
            PhotonNetwork.isMessageQueueRunning = false;
            Application.LoadLevel ("feLobby_Server");
            yield return null;
        }

        //For client, create game object.
        IEnumerator CreateLeaf() {
            string instantiateObjectShape=PhotonNetwork.player.customProperties["SHAPE"].ToString();
            //Cube, Sphere
            switch (instantiateObjectShape) {
            case PLAYERCUBE:
                PhotonNetwork.Instantiate(instantiateObjectShape,
                                          new Vector3(Random.Range(-53.0f, 53.0f),
Random.Range(26.0f, 74.0f), 25.0f),
                                          Quaternion.identity,
                                          0);
                break;
            case PLAYERSPHERE:
            default:
                PhotonNetwork.Instantiate(instantiateObjectShape,
                                          new Vector3(Random.Range(-53.0f, 53.0f),
Random.Range(26.0f, 74.0f), 30.0f),
                                          Quaternion.identity,
                                          0);
                break;
```

```
                }

                yield return null;
        }

        //When player comes to the room, add userTransformInfo.
        void OnPhotonPlayerConnected(PhotonPlayer newPlayer) {
                TransformInfo transformInfo = new TransformInfo ();
                if(userTransformInfo.ContainsKey(newPlayer.ID)==false)
                        userTransformInfo.Add (newPlayer.ID, transformInfo);
        }

        //When player leaves the room, remove userTransformInfo.
        void OnPhotonPlayerDisconnected(PhotonPlayer disconnectedPlayer) {
                if(userTransformInfo.ContainsKey(disconnectedPlayer.ID)==true)
                        userTransformInfo.Remove(disconnectedPlayer.ID);
        }

        public class TransformInfo {
                public Vector3 position;
                public Quaternion rotation;
                public Vector3 scale;
        }
}
```

(6) LeafControl.cs

```
using System.Collections.Generic;
using UnityEngine;
using System.Collections;
using System.Text;

public class LeafControl : MonoBehaviour {
        private PhotonView pv;
        private int cnt;
        private AndroidPluginManager androidPluginManager;
        private Vector3 temp_add_force_vector;
        private Rigidbody _rigidbody;
        private Transform _transform;

        private bool isServer;

        GameMgr gameMgr;
        int ownerID;

        void Start()
        {
                //PhotonNetwork sends 30 times in a second.
                PhotonNetwork.sendRate = 30;
                PhotonNetwork.sendRateOnSerialize = 30;
                pv = GetComponent<PhotonView> ();
```

```
                //All player must renew all object's texture.
                pv.RPC ("SetTextureOfPlayer", PhotonTargets.All, null);

                androidPluginManager = GetComponent<AndroidPluginManager> ();
                _rigidbody = GetComponent<Rigidbody> ();
                _transform = GetComponent<Transform> ();

                if (pv.isMine == true) {

                } else {
                        _rigidbody.isKinematic=true;
                }

                isServer = (bool)PhotonNetwork.player.customProperties ["ISSERVER"];
                GetComponent<Rigidbody> ().constraints = RigidbodyConstraints.FreezePositionZ;
                gameMgr = GameObject.Find ("GameManager").GetComponent<GameMgr> ();
                ownerID = gameObject.GetComponent<PhotonView> ().owner.ID;
        }

        void Update() {
                if (pv.isMine) {
                } else {
                        //If it is not mine, get transform data from userTransformInfo[]
                        _transform.position=Vector3.Lerp(_transform.position,
gameMgr.userTransformInfo[ownerID].position, Time.deltaTime*30.0f);
                        _transform.rotation=Quaternion.Slerp(_transform.rotation,
gameMgr.userTransformInfo[ownerID].rotation, Time.deltaTime*30.0f);
                        _transform.localScale=Vector3.Lerp(_transform.localScale,
gameMgr.userTransformInfo[ownerID].scale, Time.deltaTime*30.0f);
                }

#if UNITY_ANDROID
                //If Esc key is pressed, leave the room.
                if(Input.GetKey(KeyCode.Escape)) {
                        PhotonNetwork.LeaveRoom();
                        StartCoroutine(LoadLobby());
                }
#endif
        }

        //Leave the room.
        IEnumerator LoadLobby() {
                Destroy(gameObject);
                #if UNITY_ANDROID
                //cancel mTimer (Android)
                if (pv.isMine == true) {
                        androidPluginManager.SetCurrentObjectNameToAndroid ("");
                }
                #endif
                PhotonNetwork.isMessageQueueRunning = false;
```

```
                Application.LoadLevel ("feLobby_Client");
                yield return null;
        }

        //All player must renew object's texture when new player comes.
        [PunRPC]
        void SetTextureOfPlayer() {
                GameObject[] leaves = GameObject.FindGameObjectsWithTag ("Player");
                foreach (GameObject leaf in leaves) {
                        s       t       r       i       n       g
texture=leaf.GetComponent<PhotonView>().owner.customProperties["TEXTURE"].ToString();
                        s       t       r       i       n       g
url=leaf.GetComponent<PhotonView>().owner.customProperties["URL"].ToString();
                        //If texture exists, load it.
                        //Else, download from Amazon server.
                        if(System.String.IsNullOrEmpty(url)==true)
                        {
                                leaf.GetComponent<Renderer>().material.mainTexture=Resources.Load
(texture) as Texture;
                        }
                        else
                        {
                                StartCoroutine(DownloadTexture(leaf, url));
                        }
                }
        }

        //Download from Amazon server.
        IEnumerator DownloadTexture(GameObject leaf, string url) {
                string escapeUriString = System.Uri.EscapeUriString (url);
                WWW www = new WWW (escapeUriString);

                // Wait for download to complete
                yield return www;

                // assign texture
                leaf.GetComponent<Renderer>().material.mainTexture = www.texture;
        }

        //Synchronize object's transform.
        void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
        {
                //로컬 플레이어의 위치 정보 송신
                if (stream.isWriting) {
                        stream.SendNext (_transform.position);
                        stream.SendNext (_transform.rotation);
                        stream.SendNext (_transform.localScale);
                }
                //원격 플레이어의 위치 정보 수신
                else {
```

```
gameMgr.userTransformInfo[info.sender.ID].position=(Vector3)stream.ReceiveNext();

gameMgr.userTransformInfo[info.sender.ID].rotation=(Quaternion)stream.ReceiveNext();

gameMgr.userTransformInfo[info.sender.ID].scale=(Vector3)stream.ReceiveNext();
                    }
          }

}
```

(7) LobbyMgr.cs

```
using UnityEngine;
using System.Collections;

public class LobbyMgr : MonoBehaviour {

          // Use this for initialization
          void Start () {
                    PhotonNetwork.isMessageQueueRunning = true;
          }

          // When ESC key is pressed when player is in lobby, application is quitted.
          void Update() {
                    if(Input.GetKeyDown(KeyCode.Escape))
                    {
                              PhotonNetwork.Disconnect();
                              Application.Quit();
                    }
          }
}
```

(8) PhotonInit.cs

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class PhotonInit : MonoBehaviour {

          public string version="v1.0";
          public InstantGuiInputText userId;
          public InstantGuiButton textureSettingButton;
          public InstantGuiButton backgroundSettingButton;
          public InstantGuiButton shapeSettingButton;
          ExitGames.Client.Photon.Hashtable playerPropertyHashtable;
          ExitGames.Client.Photon.Hashtable roomPropertyHashtable;
#if UNITY_ANDROID
          AndroidJavaClass androidJavaClass;
          AndroidJavaObject currentActivity;
#endif
          private string stringBackgrounds="Backgrounds/";
```

```
	void Awake() {
		PhotonNetwork.ConnectUsingSettings (version);
		playerPropertyHashtable = new ExitGames.Client.Photon.Hashtable ();
		roomPropertyHashtable = new ExitGames.Client.Photon.Hashtable ();
#if UNITY_ANDROID
		androidJavaClass = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
		currentActivity = androidJavaClass.GetStatic<AndroidJavaObject>("currentActivity");
#endif
	}

	//On failed to connect to Photon, reconnect to server.
	void OnFailedToConnectToPhoton(DisconnectCause cause) {
		StartCoroutine (ReconnectServer ());
	}

	//Retry every 5 seconds.
	IEnumerator ReconnectServer() {
		yield return new WaitForSeconds (5);
		PhotonNetwork.ConnectUsingSettings (version);
	}

	void OnGUI() {
		//Server connection status
		GUILayout.Label (PhotonNetwork.connectionStateDetailed.ToString ());
	}

	//On Joined Lobby Callback method
	void OnJoinedLobby() {
		Debug.Log ("Entered Lobby!");
		StopCoroutine (ReconnectServer ());
		//TODO Am I Computer?

		//Initialize UserID, shape, texture.
		userId.text = GetUserID ();
		shapeSettingButton.disabled = false;
		textureSettingButton.disabled=false;
		backgroundSettingButton.disabled = false;

		//I am CLIENT
		playerPropertyHashtable["ISSERVER"] = false;
		setShape ("LeavesA", false);
		setTexture ("LeafA 1", false);
	}

	//무작위 룸 접속에 실패한 경우 호출되는 콜백함수
	void OnPhotonRandomJoinFailed() {
		Debug.Log ("No rooms!");
		//Make a room
		RoomOptions roomOptions = new RoomOptions ();
		roomOptions.customRoomProperties = roomPropertyHashtable;
```

```
        PhotonNetwork.CreateRoom ("MyRoom", roomOptions, null);
}

void OnJoinedRoom() {
        Debug.Log ("Enter Room");
        Room curRoom = PhotonNetwork.room;
        //Set roomPropertyHashtable to the current room.
        curRoom.SetCustomProperties (roomPropertyHashtable);
        StartCoroutine (this.LoadStage ());
}

//For debugging.
void OnPhotonCreateRoomFailed(object[] error) {
        Debug.Log (error [0].ToString ());
        Debug.Log (error[1].ToString());
}

//On click join room, set player's preference such as USER_ID and playerPropertyHashtable.
public void OnClickJoinRoom() {
        PhotonNetwork.player.name = userId.text;
        PlayerPrefs.SetString ("USER_ID", userId.text);
        PhotonNetwork.player.SetCustomProperties (playerPropertyHashtable);
        PhotonNetwork.JoinRandomRoom ();
}

//For UI. Texture setting.
public void OnClickLeaves(string leaf) {
        Debug.Log ("OnClickLeaves Called." + leaf);
        setTexture (leaf,false);
}

//For UI. Background setting.
public void OnClickBackground(string background) {
        Debug.Log ("OnClickBackground Called." + background);
        setBackground (background, false);
}

//For UI. Shape setting.
public void OnClickShape(string shape) {
        Debug.Log ("OnClickShape Called." + shape);
        setShape (shape, false);
}
//안드로이드로부터 받아온 이미지 파일의 경로를 저장
public void SetCustomTextureFromAndroid(string path) {
        setTexture (path, true);
}

//@Deprecated.
public void SetCustomBackgroundFromAndroid(string path) {
        setBackground (path, true);
}
```

```
	//Get user ID from text form.
	string GetUserID() {
		string userId = PlayerPrefs.GetString ("USER_ID");

		if (string.IsNullOrEmpty (userId)) {
			userId = "USER_" + Random.Range (0, 999);
		}
		return userId;
	}

	//Join the room. Load feStage.
	IEnumerator LoadStage() {
		PhotonNetwork.isMessageQueueRunning = false;
		Application.LoadLevel ("feStage");
		yield return null;
	}

	//setTexture~setShape = set user's preference through the hashtable.
	void setTexture(string leaf, bool isCustomTexture) {
		if (isCustomTexture == false) {
			playerPropertyHashtable ["URL"] = "";
			playerPropertyHashtable ["TEXTURE"] = leaf;
		} else {
			playerPropertyHashtable ["URL"] = leaf;
			playerPropertyHashtable ["TEXTURE"] = "";
		}
	}

	void setBackground(string path, bool isCustomTexture) {
		roomPropertyHashtable["ISCUSTOMBACK"] = isCustomTexture;
		roomPropertyHashtable["BACKGROUND"] = path;
	}

	void setShape(string path, bool isCustomShape) {
		if (isCustomShape == false) {
			playerPropertyHashtable ["SHAPE"] = path;
		} else {
			//TODO
		}
	}

	//If ... is clicked, load android gallery.
	public void OnClickDots(string callMethod) {
#if UNITY_ANDROID
		currentActivity.Call (callMethod, gameObject.name.ToString());
#endif
	}

}
```

(9) PhotonInitServer.cs

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

//All things are same with PhotonInit, but this is server version.
//Please check PhotonInit.

public class PhotonInitServer : MonoBehaviour {

	public string version="v1.0";
	public InstantGuiButton backgroundSettingButton;
	ExitGames.Client.Photon.Hashtable playerPropertyHashtable;
	ExitGames.Client.Photon.Hashtable roomPropertyHashtable;

	private string stringBackgrounds="Backgrounds/";

	void Awake() {
		PhotonNetwork.ConnectUsingSettings (version);
		playerPropertyHashtable = new ExitGames.Client.Photon.Hashtable ();
		roomPropertyHashtable = new ExitGames.Client.Photon.Hashtable ();
	}

	void OnFailedToConnectToPhoton(DisconnectCause cause) {
		StartCoroutine (ReconnectServer ());
	}

	IEnumerator ReconnectServer() {
		yield return new WaitForSeconds (5);
		PhotonNetwork.ConnectUsingSettings (version);
	}

	void OnGUI() {
		//Server connection status
		GUILayout.Label (PhotonNetwork.connectionStateDetailed.ToString ());
	}

	//On Joined Lobby Callback method
	void OnJoinedLobby() {
		Debug.Log ("Entered Lobby!");
		StopCoroutine (ReconnectServer ());

		//Initialize UserID, shape, texture.
		backgroundSettingButton.disabled = false;

		//I am SERVER
		playerPropertyHashtable["ISSERVER"] = true;
		setBackground (stringBackgrounds + "blue", false);
	}

	//무작위 룸 접속에 실패한 경우 호출되는 콜백함수
```

```
        void OnPhotonRandomJoinFailed() {
                Debug.Log ("No rooms!");
                //Make a room
                RoomOptions roomOptions = new RoomOptions ();
                roomOptions.customRoomProperties = roomPropertyHashtable;
                PhotonNetwork.CreateRoom ("MyRoom", roomOptions, null);
        }

        void OnJoinedRoom() {
                Debug.Log ("Enter Room");
                Room curRoom = PhotonNetwork.room;

                curRoom.SetCustomProperties (roomPropertyHashtable);
                StartCoroutine (this.LoadStage ());
        }

        void OnPhotonCreateRoomFailed(object[] error) {
                Debug.Log (error [0].ToString ());
                Debug.Log (error[1].ToString());
        }

        public void OnClickJoinRoom() {
                PlayerPrefs.SetString ("USER_ID", "SERVER");
                PhotonNetwork.player.SetCustomProperties (playerPropertyHashtable);
                PhotonNetwork.JoinRandomRoom ();
        }

        public void OnClickBackground(string background) {
                Debug.Log ("OnClickBackground Called." + background);
                setBackground (background, false);
        }

        public void SetCustomBackgroundFromAndroid(string path) {
                setBackground (path, true);
        }

        IEnumerator LoadStage() {
                PhotonNetwork.isMessageQueueRunning = false;
                Application.LoadLevel ("feStage");
                yield return null;
        }

        void setBackground(string path, bool isCustomTexture) {
                roomPropertyHashtable["ISCUSTOMBACK"] = isCustomTexture;
                roomPropertyHashtable["BACKGROUND"] = path;
        }
}
```

(10) ServerClientClassificator.cs

```
using UnityEngine;
using System.Collections;
```

```
public class ServerClientClassificator : MonoBehaviour {

	//If android, client.
	//Else, server.
	void Awake () {
		if (Application.platform == RuntimePlatform.Android) {
			Application.LoadLevel ("feLobby_Client");
		} else {
			Application.LoadLevel ("feLobby_Server");
		}
	}

}
```

(11) ServerControl.cs

```
using UnityEngine;
using System.Collections;

public class ServerControl : MonoBehaviour {
	private PhotonView pv;
	private bool isServer;

	//To drag player
	private GameObject _target;
	private bool _mouseState;

	//To make center of gravity
	private GameObject centerOfGravity = null;

	const string BLACKHOLE = "Blackhole";
	const byte EVENTCODE = 0;
	const byte EVENT_SETBLACKHOLE = 0;
	const byte EVENT_REMOVEBLACKHOLE = 1;

#if UNITY_ANDROID
	private AndroidJavaObject curActivity;
#endif
	bool isMoving=false;
	// Use this for initialization
	void Start () {
		isServer = (bool)PhotonNetwork.player.customProperties ["ISSERVER"];
		pv = gameObject.GetComponent<PhotonView> ();
		Debug.Log ("This object's ID : " + pv.owner.ID);
#if UNITY_ANDROID
		AndroidJavaClass jc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
		curActivity = jc.GetStatic<AndroidJavaObject>("currentActivity");
#endif
	}
```

```
	// Update is called once per frame
	void Update () {
		//If mouse click or touch somewhere, leaf moves to that position.
		if (isServer==true)
		{
			if(Input.GetMouseButtonDown (0)) {
				_target = GetClickedObject();

				if(_target.tag=="Player") {
					_mouseState=true;
				}
				else if(_target.tag=="Wall" && centerOfGravity==null) {
					Vector3 mousePos=Camera.main.ScreenToWorldPoint(new
Vector3(Input.mousePosition.x, Input.mousePosition.y, -Camera.main.transform.position.z));
					centerOfGravity=PhotonNetwork.Instantiate(BLACKHOLE, new
Vector3(mousePos.x, mousePos.y, 25.0f), Quaternion.identity, 0);
					SetBlackhole();
				}
			}
			else if(Input.GetMouseButtonUp(0)) {
				if(_mouseState==true) {
					_mouseState=false;
					//Animate gameobject's position randomly.
					#if UNITY_ANDROID
					pv.RPC ("Vibrate",
_target.GetComponent<PhotonView>().owner, null);
					#endif
					pv.RPC ("SetIsMoving",
_target.GetComponent<PhotonView>().owner, false);
					pv.RPC ("ScaleRestore",
_target.GetComponent<PhotonView>().owner, null);
					iTween.PunchPosition(_target, new Vector3(50.0f, 50.0f,
0.0f), 1.0f);
					iTween.ShakePosition (_target, new Vector3 (50.0f,
50.0f, 0.0f), 1.0f);
				}

				if(centerOfGravity!=null)
				{
					RemoveBlackhole();
					PhotonNetwork.Destroy(centerOfGravity);
					centerOfGravity=null;
				}
			}

			//object dragging
			if(_mouseState==true)
			{
				Vector3 mousePos=Camera.main.ScreenToWorldPoint(new
Vector3(Input.mousePosition.x, Input.mousePosition.y, -Camera.main.transform.position.z));
				pv.RPC("MoveObjectForce",
```

```
_target.GetComponent<PhotonView>().owner, mousePos);
                        }
                }

                //If object moving, object is zoomed.
                if (isServer==false && isMoving == true) {
                        if (gameObject.transform.localScale.x < 50 &&
gameObject.transform.localScale.y < 50) {
#if UNITY_ANDROID

gameObject.GetComponent<AndroidPluginManager>().isBeingMovedByServer=true;
#endif
                                gameObject.transform.localScale=Vector3.Lerp
(gameObject.transform.localScale, gameObject.transform.localScale + new Vector3 (3.0f, 3.0f, 0.0f),
                                        Time.deltaTime * 3.0f);
                        }
                }
        }

        //What is the selected object?
        GameObject GetClickedObject() {
                Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
                RaycastHit hit;
                if(Physics.Raycast(ray, out hit, 1000.0f)){
                        return hit.collider.gameObject;
                }
                return null;
        }

        //Animate Game Object
        void AnimateGameobject ()
        {
                int randomNumber=Random.Range (0, 2);

                switch (randomNumber) {
                case 0:
                        iTween.PunchPosition(_target, new Vector3(50.0f, 50.0f, 0.0f), 1.0f);
                        break;
                case 1:
                        iTween.ShakePosition (_target, new Vector3 (50.0f, 50.0f, 0.0f), 1.0f);
                        break;
                default:
                        break;
                }
        }

        //Only if it is mine, then move it.
        [PunRPC]
        void MoveObjectForce(Vector3 mousePos) {
                if (pv.isMine) {
                        gameObject.transform.position = new Vector3 (mousePos.x, mousePos.y, 25.0f);
```

```
                    SetIsMoving(true);
          }
}

//Make a blackhole. Objects are sucked into the blackhole.
void SetBlackhole() {
          byte evCode = EVENTCODE;    // my event 0. could be used as "group units"
          byte content = EVENT_SETBLACKHOLE;    // e.g. selected unity 1,2,5 and 10
          bool reliable = true;
          PhotonNetwork.RaiseEvent(evCode, content, reliable, null);
}

//Remove the blackhole.
void RemoveBlackhole() {
          byte evCode = EVENTCODE;    // my event 0. could be used as "group units"
          byte content = EVENT_REMOVEBLACKHOLE;    // e.g. selected unity 1,2,5 and 10
          bool reliable = true;
          PhotonNetwork.RaiseEvent(evCode, content, reliable, null);
}

//Vibrate phone.
[PunRPC]
void Vibrate() {
          if (pv.isMine) {
                    #if UNITY_ANDROID
                    curActivity.Call ("vibratePhone", "");
                    #endif
          }
}

//Is my object moving?
[PunRPC]
void SetIsMoving(bool value) {
          if (pv.isMine) {
                    isMoving = value;
          }
}

//If mouse click ends, restore the scale of object.
[PunRPC]
void ScaleRestore() {
          if (pv.isMine) {
                    gameObject.transform.localScale=new Vector3(10.0f, 10.0f, 10.0f);
                    #if UNITY_ANDROID
                    gameObject.GetComponent<AndroidPluginManager>().isBeingMovedByServer=false;
                    #endif
          }
}

// setup our OnEvent as callback:
void Awake()
```

```
	{
		PhotonNetwork.OnEventCall += this.OnEvent;
	}

	// handle events.
	private void OnEvent(byte eventcode, object content, int senderid)
	{
		if (eventcode == 0)
		{
			byte selected = (byte)content;
			Debug.Log ("OnEvent!!!");
			if(pv.isMine)
			{
				switch(selected)
				{
				case EVENT_SETBLACKHOLE:
					Debug.Log ("EVENT_SETBLACKHOLE called!!!");
					GameObject objBlackhole =
GameObject.FindGameObjectWithTag (BLACKHOLE);

gameObject.GetComponent<FauxGravityBody>().attractor=objBlackhole.GetComponent<FauxGravityAttractor>();
					break;
				case EVENT_REMOVEBLACKHOLE:
					Debug.Log ("EVENT_REMOVEBLACKHOLE called!!!");
					Vibrate ();

gameObject.GetComponent<FauxGravityBody>().attractor=null;
					break;
				default:
					break;
				}

			}
		}
	}
}
```

3. 이미지 서버 php

(1) uploadtoserver.php

```
<?php
    header("Content-Type: multipart/form-data; charset=UTF-8");
    $file_path="uploads/";
    $file_name=$_FILES['uploaded_file']['name'];
    //if  (($_FILES["uploaded_file"]["type"]  ==  "image/png")  &&  ($_FILES["uploaded_file"]["size"]  <
20000000000))
    //{
      if ($_FILES['uploaded_file']['error'] > 0)
      {
         echo "Return Code: " . $_FILES['uploaded_file']['error'] . "";
      }
      else
```

```
      {

         echo "==>".var_dump(iconv_get_encoding('all'))."";
         echo "Upload: " . $_FILES['uploaded_file']['name'] . "";
         echo "Type: " . $_FILES['uploaded_file']['type'] . "";
         echo "Size: " . ($_FILES['uploaded_file']['size'] / 1024) . " Kb";
         echo "Temp file: " . $_FILES['uploaded_file']['tmp_name'] . "";

         if (file_exists($file_path . $_FILES['uploaded_file']['name']))
         {
             echo $_FILES['uploaded_file']['name'] . " already exists. ";
         }
         else
         {
              //move_uploaded_file($_FILES['uploaded_file']['tmp_name'],            $file_path            .
$_FILES['uploaded_file']['name']);
              move_uploaded_file($_FILES['uploaded_file']['tmp_name'], $file_path . $file_name);
              echo "Stored in: " . $file_path . $file_name;
         }
      }
    //}
    //else
    //{
    //  echo "Invalid file";
    //}
    //phpinfo();
?>
```