3D modeling with the ThreeJS Framework of the Colonial Temple of Chuquinga Aymaraes – Perú

Michael A. Pimentel Palomino, Ecler Mamani Vilca and Wilson Mollocondo Flores

Abstract - This research has the purpose of modeling in 3D of the Colonial Temple of Chuquinga, to prove the rendering time in mobile devices, the investigation is applied, with an explanatory level and an experimental design where the rendering time with the frequency was evaluated of images (FPS, obtaining the frequency of images (FPS), the average is 52.29 FPS, highlighting the most influential GPU, that is, increases the frequency of images (FPS) when the central clock is greater than 500 Mhz in all mobile devices that have an optimum level, while in basic level devices when GPM Mhz decreases, the frequency of images (FPS) decreases, however, it does not cause distortion in animations and textures. The ThreeJS library was used to create 3D objects, 3ds Max for the modeling of temple walls and WebGL for final rendering.

Keywords: 3D Modeling, ThreeJS, Colonial Temple, Chuquinga, Aymaraes

1 INTRODUCCIÓN

El modelado en 3D es la representación matemática de un objeto tridimensional el cual se puede visualizar mediante la renderización, por otro extremo se tiene la preservación del patrimonio arquitectónico de la nación [1] y la sinergia de plasmar la interculturidad en el desarrollo del software educativo [2], son los motivos para realizar el Modelado en 3D del Templo Colonial de Chuquinga ubicado en la provincia de Aymaraes, Región de Apurímac, Perú , utilizando Framework ThreeJS los cuales deben son visualizados (renderizados) en los dispositivos móviles .

En el modelado 3D se describe los métodos, conjunto de características formado por objetos poligonales, tonalidades, texturas, sombras, reflejos, transparencias, translucidez, refraxiones, iluminación (directa, indirecta y global), profundidad de campo, desenfoques por movimiento, ambiente cuyo resultado será un simulador de recorrido en 3D. Así también se evalúa que componentes de los dispositivos móviles tiene una relación directa en la renderización. Actualmente la visualización de los modelados 3D en navegadores Web tienen dificultades en el control del diseño por parte de los desarrolladores[3], realizó una valoración cualitativamente con distintas algunas de las cuales ha permitido extraer sus fortalezas y debilidades el resultado final fue la explicación de lo que ocurre internamente y cómo se relacionan las clases en cada caso de uso en los gráficos, la cantidad de fotogramas por segundo en una renderización influye directamente presentación interactiva de escenas en Android limitando inicialmente a la tasa de representación que se debe alcanzar como mínimo de diez frames por segundo para plataformas Android[4] para lo cual se realizó la portabilidad de la librería OSG con script de NDK añadiendo macros en CNAKE para poder ser empleado en una arquitectura externa representado escenas complejas con una tasa de dibujo interactiva, teniendo una serie de cambios en los archivos del código fuente de la librería para distribuciones de Linux y Android. La inexistencia de una técnica única para generar imágenes tridimensionales en un sistema de visualización médica permitió el e diseñó de una metodología para la evaluación entre técnicas de renderizado 3D, concluyendo en la comparación entre técnicas de rendering para imágenes diagnóstico, como lo fueron en este caso Matlab y C# debido a esto se encontró que la técnicas de rendering de volumen ofreciendo una mejor visualización desarrollado en C# es la buena calidad de imágenes poco precisas ya que solo muestra el exterior de la imagen [5], Otra limitación de los dispositivos móviles es la potencia de cálculo por tamaño de la memoria para acelera dicho proceso se crean la técnicas de la multiresolución mediante imágenes precalculadas para sustituir geometría y obtener efectos realistas a bajo coste visualización un conjunto de puntos sobre la superficie de los objetos y visualización basada en polígonos [6], obteniéndose un número de imágenes por segundo que varía mucho en las escenas por lo que se realizan muchos envíos de geometría, de ahí las algunas imágenes que aparecen en la gráfica [6], a lo mencionado de las diferentes investigaciones se demuestra en la portabilidad de la arquitectura obteniéndose el máximo rendimiento del dispositivo móvil al estimar el soporte máximo de FPS acotando a ello que el modelado no solo es una imagen, es un recorrido virtual por las instalaciones del templo.

En un pequeño pueblo de la provincia apurimeña de Aymaraes, ubicado en la margen izquierda del rio Chalhuanca a 2870 m.s.n.m. luego de cruenta batalla en Chuquinga, del 21 de mayo de 1554, en los dos siglos, Chuquinga fue el punto de partida para el ascenso a la mina de oro y asentamiento minero de Huaillaripa en las punas de la comunidad. Durante el auge de la ex-plotación de las minas en el siglo XVII fue construido el hermoso templo de Huaillaripa, ahora en ruina, y la Iglesia de Chuquinga, posiblemente en remplazo con una edificación más rustica [7].

Fig. 1: Templo Colonial de Chuquinga [7], bajo la advocación de San Pedro, presenta una planta de distribución longitudinal, con tres volúmenes adosados al cuerpo principal, en el muro lateral sur, correspondiente a los ambientes del baptisterio, la capilla y la sacristía, lo cual se reconstruirá virtualmente [8].

Los tejados contienen pinturas de forma curva y tamaño relativamente uniforme, mide entre 45 y 50 cm de largo, 21 a 24 cm en el extremo angosto y la curvatura de 9 a 12 cm, existen diferentes diseños como se pueden observar en las figuras 2, 3 y 4.

- Michael Alexander Pimentel Palomino, Escuela profesional de Informática y Sistemas, UNAMBA – Perú, michael101136@gmail.com
- Ecler Mamani Vilca, Escuela profesional de Informática y Sistemas, UNAMBA – Perú, eclervirtual@unamba.edu.pe
- Wilson J. Mollocndo Flores, Departamento de Ciencias, UNAMBA – Perú, wilsonmollocondo@unamba.edu.pe

Fig. 2. Tejado con gráfico de cocos.

Fig. 3. Tejado con gráfico de lagartijas

Fig. 4. Tejado con re prestaciones andinas, similares al templo de San Blas de la Cuidad de Cusco, Perú.

2 METODOLOGÍA

El método inductivo porque permitió describir, detallar, especificar todos los aspectos del desarrollo del modelado 3D del Templo Colonial de Chuquinga con el Framework ThreeJS, con diseño experimental muestra es no probabilística [8], los dispositivos móviles debe de ser: originales, superiores o iguales a gama baja, sean de la marca Samsung, LG, Sony, Nokia, Motorola, Huawei, Bitel, Mobile, Doogee, HTC y Xioami en una cantidad de n=30. La observación siendo la técnica para obtener la información de tiempo de renderización y frecuencia de imágenes (FPS), como instrumentos de recolección de datos se usó el registro del tiempo de renderización y frecuencia de imágenes (FPS) de los dispositivos móviles.

Dos fases se contaron para el desarrollo de la aplicación:

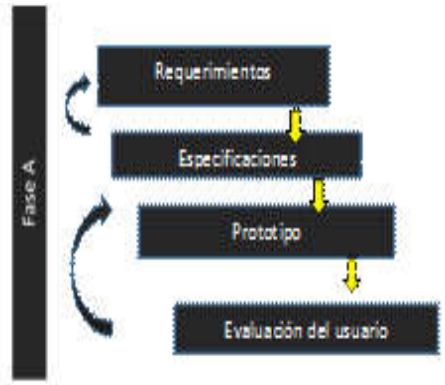

Fig. 4. Fases del desarrollo del Software.

3 RESULTADOS

Sobre el modelado, se crearon os objetos y elementos que conforman el modelado el templo se usó la herramienta 3ds Max utilizado figuras geométricas complejas, como sillas, puerta de entrada y los muros del templo. La ubicación y especificaciones técnicas nos basamos en el plano catastral Chalhuanca 2004 Fig. 5.

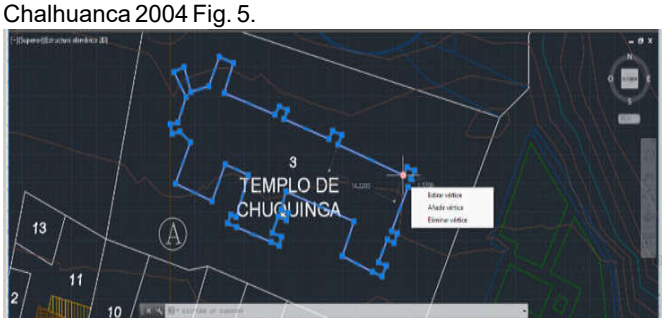

Fig. 5. Plano catastral de Chalhuanca [9]

La escala (metros), las medidas de la Fig. 6 son el contorno del templo aproximado de 191.49 metros, el grosor de las paredes 0.031 metros, el baptisterio 7.17, bautisterio 8.69 metros, y el bautisterios 6.39 metros.

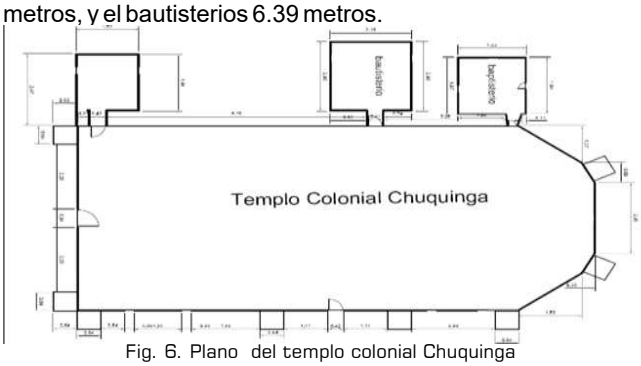

Fig. 6. Plano del templo colonial Chuquinga

La técnica utilizada fue la editable poly, consiste en modificar los polígonos a través de vértices, aristas y borde basado en una malla editable usando pocos polígonos, para que el dispositivo móvil lo procese de forma rápida. Para la recreación del ambiente se realizó: creación del plano, modelado, texturizado a partir de fotos tomadas del templo.

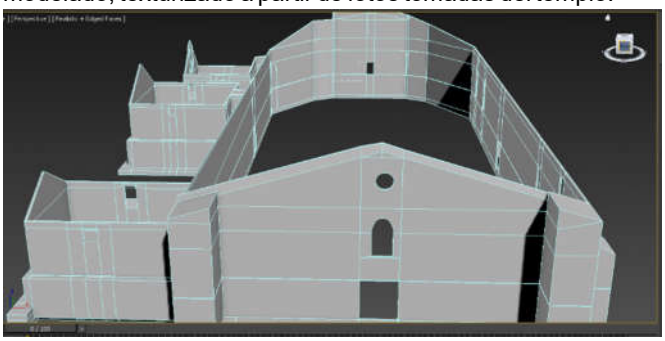

Fig. 7. Modelado de los contornos y techos del templo.

Para el modelado de las objetos internos fue creado partir de una foto tomada en el templo y utilizando el modificador lathe de 3ds Max como se muestra en la Fig 7., para después reducir las aristas y vértices del polígono sin que pierda el detalle del objeto.

Fig. 8. Modelado de la columna de soporte del segundo piso.

ISSN 2706-543X - Ing. Informática y Sistemas - UNAMBA

Para el texturizado se utilizó la técnica UVW mapping, permite pintar las superficie en base a una textura sólida.

Fig. 9. Texturizado del Arco Interno

Sobre la navegación del escenario se utilizó la librería ThreeJS, es liviana y eficiente para generar y animar gráficos en 3D dentro del navegador, aprovechando las grandes novedades que nos ofrece HTML5 para la generación de contenidos multimedia. Las ventajas de esta no tienen ninguna dependencia a otras bibliotecas sin embargo, para aprovechar ThreeJS al máximo, necesita un navegador que admita el estándar WebGL [10].

Tabla 1: Compatibilidad de navegadores

Navegador	Admite WebGL
Internet Explorer	Versión 11.
Mozila Firefox	Versión 4.
Google Chrome	versión 10
Safari	Versión 5.1 y superiores .
Opera	WebGL desde la versión 12.0.

En la estructura se programó: escena, cámara y renderizador. La cámara utilizada fue PerspectiveCamera como se observa la codificación en la figura Fig. 10.

```
renderer = new THREE.WebGLRenderer( { antialias: true } );
renderer.setPixelRatio( window.devicePixelRatio );
renderer.setSize( window.innerWidth, window.innerHeight );
container.appendChild( renderer.domElement );

scene = new THREE.Scene();

camera = new THREE.PerspectiveCamera( 45, window.innerWidth / window.innerHeight, 1, 10
camera.position.x = -400;
camera.position.z = 4000;
camera.position.y = 300;
```

Fig. 10. Extracción de la codificación del PerspectiveCamera.

La renderización en ThreeJS lo realiza con Canvas Render, la cual diseña sus objetos 3D usando WebGL Render.

```
nction init() {

renderer = new THREE.WebGLRenderer( { antialias: true } );
renderer.setPixelRatio( window.devicePixelRatio );
renderer.setSize( window.innerWidth, window.innerHeight);//renderer.
window.innerHeight/2,false );

container.appendChild( renderer.domElement );

scene = new THREE.Scene();
```

Fig. 11. Extracción de la codificación para la renderización.

En cada una de los elementos se trabajó con aristas y vértices para no sobrecargar el aplicativo de los modelados a continuación se muestra el código la cual genera las gradas y niveles de un costado de la entrada del templo.

```
var Geometria=new THREE.Geometry();
var vertices=[[2,2,0],[70,2,0],[70,35,0],[35,35,0],[35,70,0],[2,70,0]];
var long_vertices=vertices.length;
var array_extrude=[];
for(i=0;i<long_vertices;i++){
        x=vertices[i][0];
        y=vertices[i][1];
        z=vertices[i][2];
        Vector=new THREE.Vector3(x,y,z);
        Geometria.vertices.push(Vector);
        array_extrude.push(Vector);
}
var forma_figura=new THREE.Shape(array_extrude);
var datos_extrusion={
        amount:500,
        bevelEnabled:false,
        bevelSegments:1,
        steps:5,
        bevelThickness:100
        };
var textura=THREE.ImageUtils.loadTexture('texturas/piedra/piedra.jpg' );
var extrude_geometria=new THREE.ExtrudeGeometry(forma_figura,datos_extrusion);
textura.repeat.set(0.02,0.02);
textura.wrapS = textura.wrapT = THREE.repeatWrapping;
var material = new
THREE.MeshBasicMaterial({map:textura,side:THREE.DoubleSide,wireframe:false});
var mallaextrusion=new THREE.Mesh(extrude_geometria,material);
mallaextrusion.position.set(-1000,-19,1440);
```

La ejecución del código se observa en la Fig. 12.

Fig. 12. Renderizado de las gradas de la entrada al Templo.

Para la renderización de los paisajes, se utilizó el formato de panorama definiéndose como la proyección utilizada para mapear la escena 3D total o parcial basado en el formato utilizando 6 caras para llenar toda la esfera que rodela al templo. La imagen se asigna a las caras de los cubos que se ajustan perfectamente, a continuación se muestra el código fuente la cual genera el cielo utilizando la proyección cúbica.

```
var geometry = new THREE.CubeGeometry( 120000, 25000, 120000 );
var cubeMaterials =
        [
new THREE.MeshBasicMaterial( { map: new THREE.TextureLoader( ).load( 'texturas/Skybox/front.jpg' ), side: THREE.DoubleSide } ),
new THREE.MeshBasicMaterial( { map: new THREE.TextureLoader( ).load( 'texturas/Skybox/back.jpg' ), side: THREE.DoubleSide } ),
new THREE.MeshBasicMaterial( { map: new THREE.TextureLoader( ).load( 'texturas/Skybox/up.jpg' ), side: THREE.DoubleSide } ),
new THREE.MeshBasicMaterial( { map: new THREE.TextureLoader( ).load( 'texturas/Skybox/down.jpg' ), side: THREE.DoubleSide } ),
new THREE.MeshBasicMaterial( { map: new THREE.TextureLoader( ).load( 'texturas/Skybox/right.jpg' ), side: THREE.DoubleSide } ),
new THREE.MeshBasicMaterial( { map: new THREE.TextureLoader( ).load( 'texturas/Skybox/left.jpg' ), side: THREE.DoubleSide } )
        ];
var cubeMaterial = new THREE.MeshFaceMaterial( cubeMaterials );
var cube = new THREE.Mesh( geometry, cubeMaterial );
cube.position.set(-910,3940,1400); scene.add( cube );
```

La ejecución del código se observa en la fig. 13.

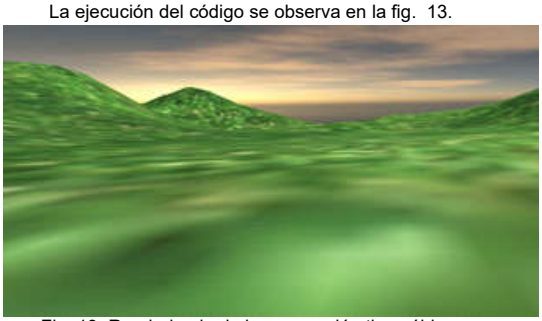

Fig. 13. Renderizado de la proyección tipo cúbico.

Sobre el resultado de la integración reunir todos objetos modelados en un mismo escenario utilizando el Framework ThreeJS para ello se utilizó el complemento AssimpJSONLoade de ThreeJS con el que se cargó modelos creados en 3ds Max, el complemento AssimpJSONLoade acepta archivos en formato JSON para cual se utilizó el exportador ThreeJSExport versión v0.8, los resultados se observan en las figuras 14, 13 y 15.

Fig. 14. Renderizado en formato JSON e integradas a ThreeJS.

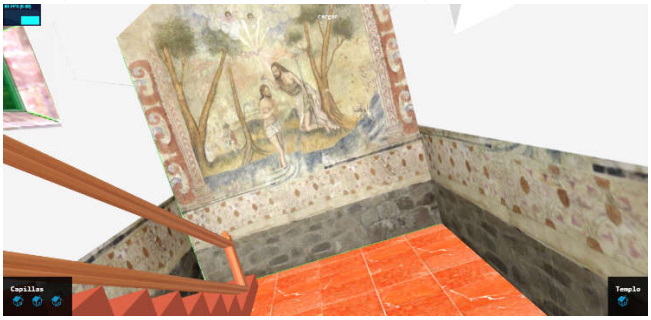

Fig. 14. Renderizado Primera capilla del Templo Colonial de Chuquinga.

Fig. 15. Vista a través de las ventanas, visualización desde el exterior del templo

A continuación mostramos los resultados para el tiempo de renderización en dispositivos móviles se determinaron los factores:

Tabla 2: Factores y rangos de los compontes básicos de los dispositivos móviles.

FACTOR	TIPO	RANGO
Memoria interna	Controlable	[16GB < , = 16GB]
Memoria RAM	Controlable	[2GB < , = 2GB]
GPU	Controlable	[500 MHZ= , >500 MHZ]
Internet	No controlable	

Los resultados obtenidos mediante el del análisis de varianza del tiempo de renderizado.

Tabla 3: Análisis de varianza del tiempo de renderizado

Promedios del tiempo de renderizado	
Memoria interna (α_i)	0.4921
Memoria RAM (β_j)	0.4011
GPU(Υ_K)	-1.6630

Los resultados de la estimación de efectos promedios mostrados en la tabla 4, se obtuvieron procesando tiempo de renderización registrados en cada dispositivo móvil.

Tabla 4: Promedios para el tiempo de renderizado

Fuente	SC	Df	MC	F	P-Value
Memoria interna (α_i)	2.7037	1	0.54818	0.89	0.354
Memoria RAM (β_j)	0.0240	1	0.62825	1.02	0.322
GPU (Υ_K)	5.4451	1	5.74865	9.36	0.005

La memoria interna y la memoria RAM afectan en el tiempo de renderizado debido a que sus pendientes son pequeñas por los cual se deberán evaluar en el diagrama de Pareto y la gráfica de efectos estandarizados para ver si estos dos factores afectan significativamente o no en el tiempo de renderizado, asi también se muestra en la Fig. 16.

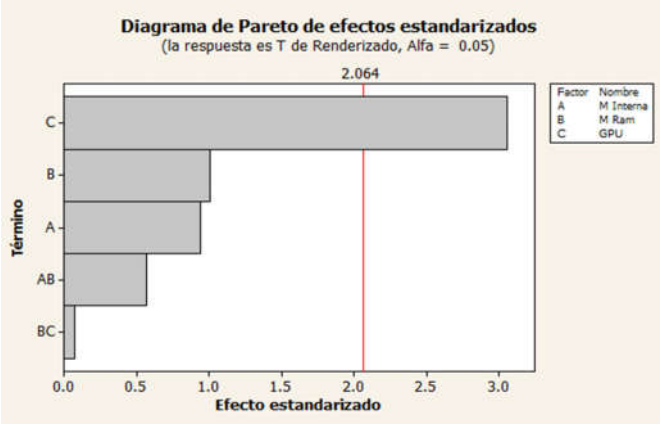

Fig. 16 : Diagrama de Pareto de efectos estandarizados.

En la figura N° 16, las barras que cruzan la línea de referencia son significativas es decir la barra C la cual es el factor GPU es el único que cruza la línea de referencia. Este factor es estadísticamente significativo en el nivel de 0.05.

4 CONCLUSIONES

Se modeló en 3D el templo de Chuquinga, utilizando el Framework ThreeJS para crear objetos 3D, 3ds Max para el modelado de los muros del templo y WebGL para el renderizado final como una aplicación web, independiente de la arquitectura del dispositivo de móvil.

Para el tiempo de renderización en dispositivos móviles del modelado 3D, el factor GPU, existe un efecto principal en el factor GPU en la gráfica de efectos indica cuando la frecuencia del reloj central es mayor a 500 Mhz, reduce el tiempo de renderización, también observó que la GPU supera la línea de referencia en el diagrama de Pareto, indicando que este factor es significativo y en el análisis de varianza con tres factores se obtuvo el p-value=0.005, al ver que el valor de probabilidad es menor al nivel de significancia (0.005<0.05) lo que confirma que la GPU influye significativamente en el tiempo de renderizado del modelo 3D del Templo Colonial de Chuquinga con ThreeJS.

AGRADECIMIENTOS

A la municipalidad de Chalhuanca por facilitarnos el archivo del catastro y el acceso al Templo para realizar los requerimientos para el modelado.

REFERENCIAS

[1] C. d. l. R. d. Perú, Ley General del Patrimonio Cultural de la Nación LEY Nº 28296, Lima: Congreso de la República del Perú, 2016.

[2] E. MAMANI VILCA, "Propuesta de un enfoque para el desarrollo de software educativo intercultural," 3er Simposio Internacional de Innovación y Tecnología ISIT2012, vol. 2012, pp. 152-158, 05 12 2012.

[3] J. V. Pereira, Gráficos 3D en Interfaces Web, Tenerife: Universidad San Cristóbal de la Laguna, 2014.

[4] J. I. Cige, "Visualización interactiva de escenas tridimensionales con OpenSceneGraph sobre dispositivos Android," Universidad Politécnica de Valencia, España, 211.

[5] G. María Quinteros, "Metodología para la evaluación de técnicas de renderizado 3D en sistemas de visualización de imágenes médicas," Universidad Tecnológica de Pereira, Colombia, 2014.

[6] E. MAMANI VILCA, J. . C. MUÑOZ MIRANDA and F. N. TUMI, "HITO: Pseudocode compiler with graphics library," *5TH International Symposium on Innovation and Technology,* pp. 88-92, 2016.

[7] R. Gaitán, "Visualización interactiva 3D en dispositivos móvi," Universidad Politécnica de Valencia, Valencia, 2015.

[8] R. Hostnig, Boletin de Lima, Lima: El Pino, 2006.

[9] G. Jiménez and F. Mendoza, "Animación y reconstrucción sobre la zona Prehispánica de Monte Albán, Oaxaca," Universidad Nacional Autónoma de México, CDMX, 2014.

[10] R. Hernandez Sampieri, F. C. Carlos and P. L. Baptista, Metodología de la investigación, México: Interamericana Editores, 2014.

[11] M. Challhuanca, Plano Catastral, Chalhuanca, 2014.

[12] J. Dirksen, Learning Three.js: TheJavaScript 3D Library for WebGL, UK: Packt Open source, 2015.

BIOGRAFÍAS

Michael Alexander Pimentel Palomino, Ingeniero Informático y Sistemas egresado de la Universidad Nacional Micaela Bastidas de Apurímac. Experiencia en Desarrollo de software, base de datos, manejo de programas de diseño 3D y realidad aumentada, actualmente webmaster en la empresa de turismo Tierras de los Andes S.A.C – Cusco.

Ecler Mamani Vilca, Graduado en la Universidad Nacional del Altiplano 2002 Ingeniero Estadístico, Informático 2001– Maestro en Educación 2008 - Perú, experiencia en desarrollo de aplicaciones multimedia, innovador tecnológico e investigador en informática, docente de la Universidad Nacional Micaela Bastidas de Apurímac – Perú en la Escuela Profesional de Ing. Informática y Sistemas. Miembro activo del Instituto de Investigación en Ciencias de la Computación – UNAP, Perú

Wilson J. Mollocondo Flores, Doctor en Administración, Magister Scientie en informática obtenido en la Universidad Nacional del Alti-plano -Puno, actualmente docente Principal del departamento de Ciencias Básicas de la Universidad Nacional Micaela Bastidas de Apurímac Perú.