Tecnológico Nacional de México

Instituto Tecnológico de Oaxaca

Ingeniería en Sistemas Computacionales

Reporte Proyecto de Unidad IV - V

GRAFICACIÓN

Docente: Martínez Nieto Adelina

Alumnos:

Santos Santos Mark Ángel

Luis David Porcallo Hernández

Grupo: 6V1

Unidad I

Contenido

Introducción

El proyecto de evaluación de la unidad cuatro y cinco de la materia de Graficación es un emocionante desafío que nos ha permitido aplicar los conocimientos adquiridos durante las últimas tres semanas de estudio en un contexto práctico y creativo. Nuestro objetivo es desarrollar un programa en el lenguaje de programación Java que simule las impresionantes ruinas arqueológicas de Monte Albán, un sitio histórico y culturalmente significativo ubicado en el estado de Oaxaca.

Para llevar a cabo esta tarea, hemos explorado diversas técnicas de animación 2D y 3D utilizando las librerías de Java2D y Java3D. Aunque no contamos con una librería específica para estos diseños, hemos aprovechado el poder de softwares de modelado 3D como Sketchup o Blender para crear una representación detallada y realista de las ruinas. Esta combinación de tecnologías nos ha permitido lograr un resultado visual impresionante y enriquecedor.

El proyecto no se trata solo de programación, sino también de sumergirse en la historia y la cultura de Monte Albán. Hemos investigado a fondo para asegurarnos de que nuestra simulación refleje con precisión la majestuosidad y el encanto de este importante lugar histórico. Cada detalle cuenta, desde la arquitectura de las estructuras hasta los artefactos antiguos que encontramos en la zona. Queremos que los usuarios del programa se sientan transportados a través del tiempo mientras exploran estas ruinas virtuales.

Desarrollo

Diseño

Diseño del modelado

A través de este paso, nos basamos en las diferentes imágenes que podemos obtener de Google maps para la construcción del modelado, en este caso nos solicitan 4 piramides de las ruinas:

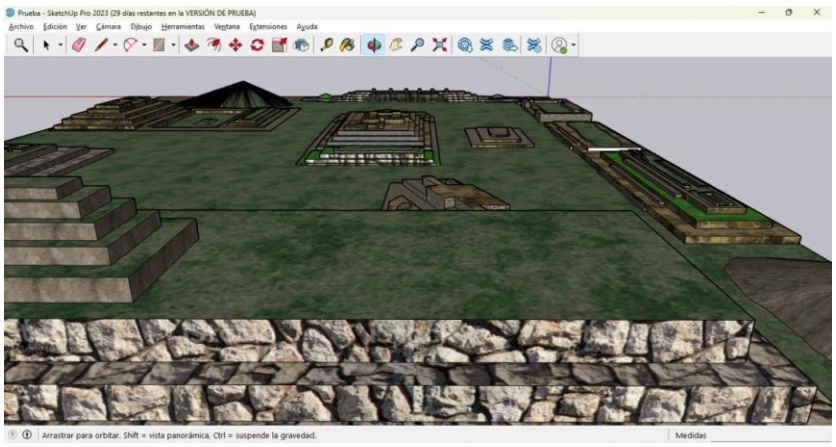
La Gran Plaza: Aunque técnicamente no es una pirámide en sí misma, la Gran Plaza es el corazón del sitio arqueológico de Monte Albán y es donde se encuentran las otras estructuras destacadas. Aquí, en el centro de la plaza, hay una explanada abierta que habría sido utilizada para ceremonias y eventos importantes.

La Pirámide de los Danzantes: Esta es una de las estructuras más conocidas y distintivas de Monte Albán. Se le llama "de los Danzantes" debido a los frisos tallados que se encuentran en sus lados. Estas figuras talladas representan figuras humanas en posturas contorsionadas y, aunque su significado exacto sigue siendo un misterio, se cree que pueden estar relacionadas con rituales, danzas o aspectos religiosos de la cultura zapoteca.

La Pirámide de la Luna: Situada en el lado oeste de la Gran Plaza, esta pirámide debe su nombre a una escultura de piedra que se asemeja a un creciente lunar encontrada en su cima. Esta pirámide es más pequeña que la Pirámide del Sol, pero aún así es impresionante y muestra la habilidad arquitectónica de los zapotecas.

La Pirámide del Sol: Situada al sur de la Gran Plaza, esta pirámide es la más grande de Monte Albán y una de las estructuras más significativas del sitio. Fue construida como una plataforma funeraria y se cree que albergaba las tumbas de gobernantes o personajes importantes de la época zapoteca. Es una estructura imponente y ofrece vistas impresionantes del paisaje circundante.

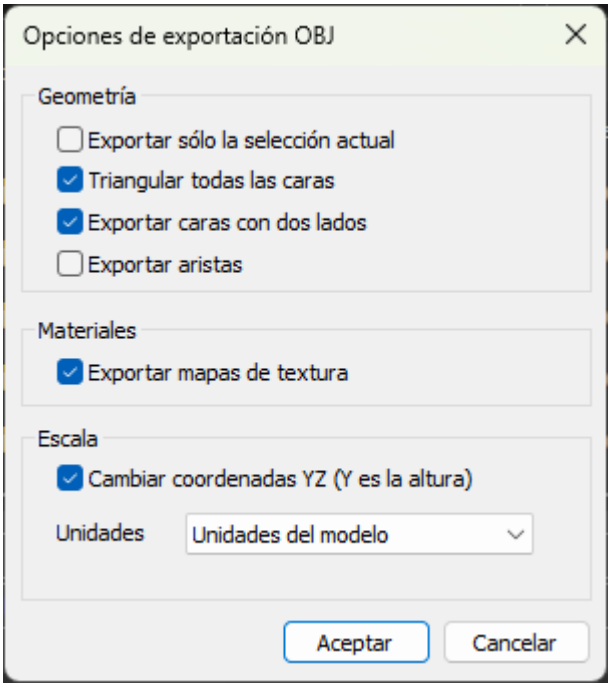
Sin embargo, nos animamos a diseñar la mayor parte de las ruinas (sin contar la parte de la entrada de estas), a través del programa Sketchup, que al final este fue el resultado:



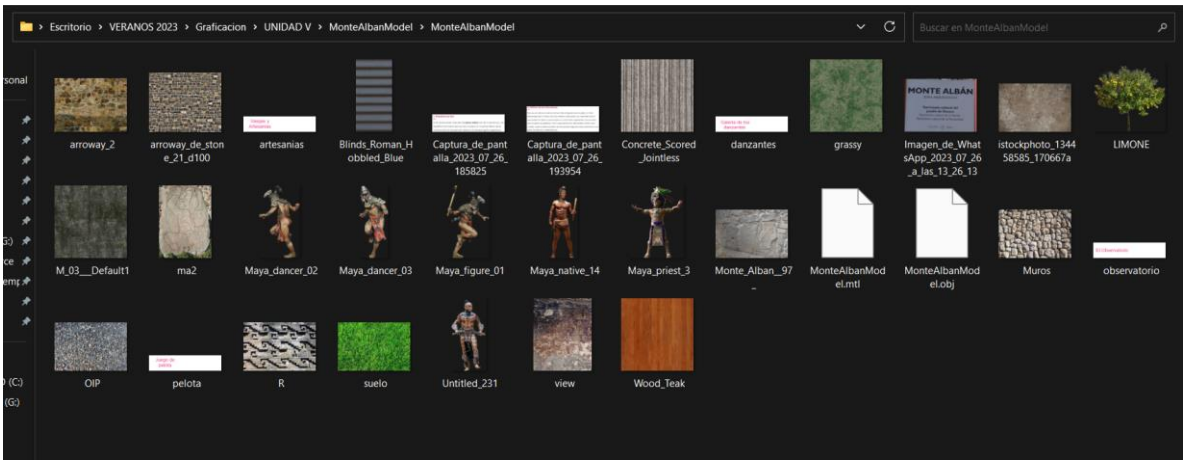
Este modelo fue hecho desde cero, solo nos basamos en otros modelos y en fotografías del lugar.

Exportación del modelo

Para la exportación del modelo, solo guardamos nuestro proyecto y damos clic sobre el botón que dice archivo y nos vamos hasta la sección exportar, luego elegimos el modelo 3D, y seleccionamos la ubicación de la carpeta, solo que antes de seguir, abrimos las opciones y seleccionamos estas casillas:



Después comenzamos con la exportación y al final nos debe dar una carpeta con lo siguiente:

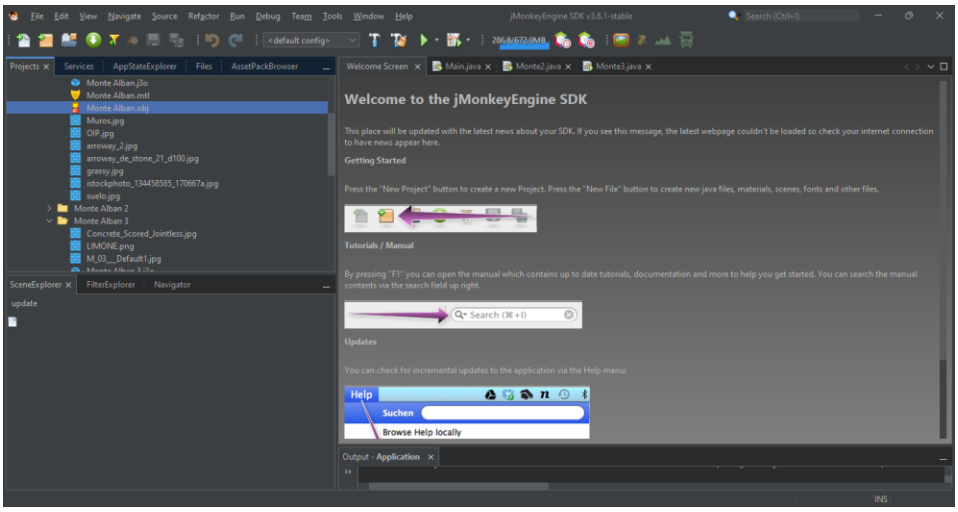


Importación del modelo al JMonkey

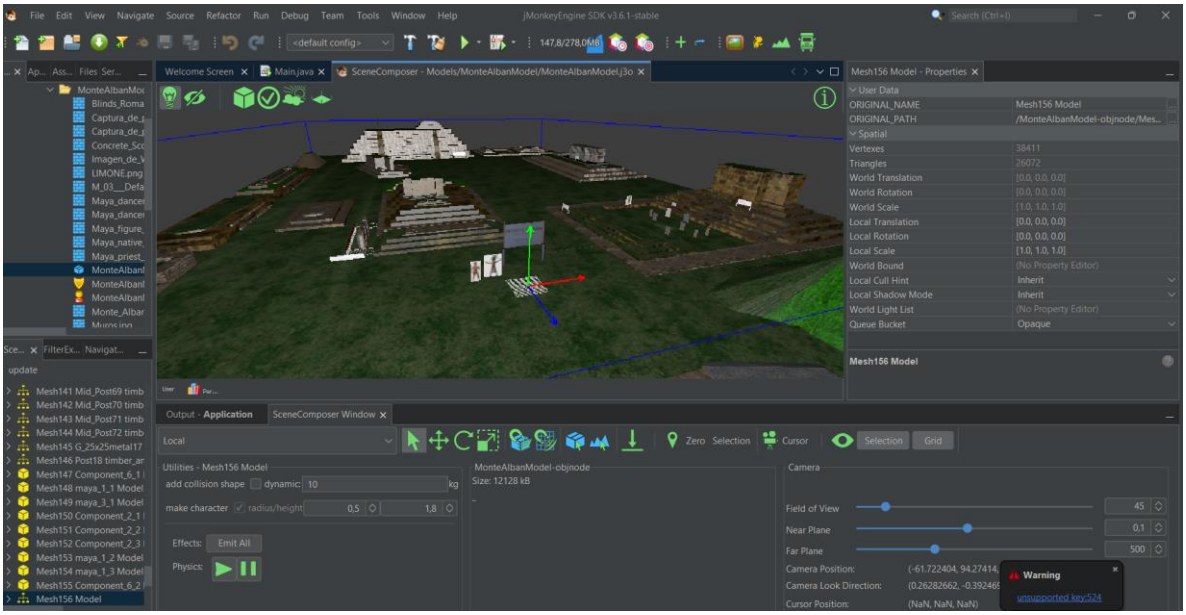
Este es lo mas importante del proyecto, ya que abrimos el jmonkey y creamos un nuevo proyecto (que tiene que ser jm3 y de ahí un basic game with ant) es debido a que tendrán las librerías jm3 para ejecutar el programa (es como un editor de videojuegos).

Ya que creamos nuestro proyecto, seleccionamos nuestra carpeta que creamos al exportar el modelo y lo movemos hacia la carpeta Models.

Ya que este cargado hacemos lo siguiente:



Damos clic derecho sobre el archivo .obj, y le decimos que lo convierta a .j3o, que al abrirlo nos dará esto:



Nos aparecerá nuestro modelo 3d de las ruinas.

Codificación

Ahora solo faltaría programar nuestro personaje para que podamos andar sobre ella, y es con el siguiente código:

```
  private void setUpKeys() {
    inputManager.addMapping("Left", new
KeyTrigger(KeyInput.KEY_A));
    inputManager.addMapping("Right", new
KeyTrigger(KeyInput.KEY_D));
    inputManager.addMapping("Up", new
KeyTrigger(KeyInput.KEY_W));
    inputManager.addMapping("Down", new
KeyTrigger(KeyInput.KEY_S));
    inputManager.addMapping("Jump", new
KeyTrigger(KeyInput.KEY_SPACE));
    inputManager.addListener(this, "Left");
    inputManager.addListener(this, "Right");
    inputManager.addListener(this, "Up");
    inputManager.addListener(this, "Down");
```

```
    inputManager.addListener(this, "Jump");
  }
  /** These are our custom actions triggered by key presses.
   * We do not walk yet, we just keep track of the direction the user
pressed. */
  public void onAction(String binding, boolean value, float tpf) {
    if (binding.equals("Left")) {
      if (value) { left = true; } else { left = false; }
    } else if (binding.equals("Right")) {
      if (value) { right = true; } else { right = false; }
    } else if (binding.equals("Up")) {
      if (value) { up = true; } else { up = false; }
    } else if (binding.equals("Down")) {
      if (value) { down = true; } else { down = false; }
    } else if (binding.equals("Jump")) {
      player.jump();
    }
  }
```

Estos dos métodos son para que nuestro personaje pueda rotar cámara y caminar sobre el modelo.

Y para cargar el modelo es la siguiente:

```
@Override
   public void simpleInitApp() {
      DirectionalLight sun = new DirectionalLight();
      sun.setColor(ColorRGBA.White.mult(0.0f));
      sun.setEnabled(true);
      rootNode.addLight(sun);
      flyCam.setMoveSpeed(20);

      Spatial casona;
      casona                                                    =
assetManager.loadModel("Models/MonteAlbanModel/MonteAlbanModel.j3o");
      casona.setLocalScale(2f);
      rootNode.attachChild(casona);
```

```
        /**
         * Set up Physics
         */
        bulletAppState = new BulletAppState();
        stateManager.attach(bulletAppState);
        //bulletAppState.getPhysicsSpace().enableDebug(assetManager);
        // We re-use the flyby camera for rotation, while positioning is handled by
physics
        viewPort.setBackgroundColor(new ColorRGBA(0.7f, 0.8f, 1f, 1f));
        flyCam.setMoveSpeed(100);
        setUpKeys();
        setUpLight();
        // We set up collision detection for the scene by creating a
        // compound collision shape and a static RigidBodyControl with mass zero.
        CollisionShape                    sceneShape                    =
CollisionShapeFactory.createMeshShape((Node) casona);
        landscape = new RigidBodyControl(sceneShape, 0);

        casona.addControl(landscape);
        // We set up collision detection for the player by creating
        // a capsule collision shape and a CharacterControl.
        // The CharacterControl offers extra settings for
        // size, stepheight, jumping, falling, and gravity.
        // We also put the player in its starting position.
        CapsuleCollisionShape capsuleShape = new CapsuleCollisionShape(0.4f, 2f,
1);
        player = new CharacterControl(capsuleShape, 0.4f);
        player.setJumpSpeed(30);
        player.setFallSpeed(500);
        player.setGravity(30);
        player.setPhysicsLocation(new Vector3f(0, 10, 0));
        // We attach the scene and the player to the rootnode and the physics space,
        // to make them appear in the game world.
        rootNode.attachChild(casona);
        bulletAppState.getPhysicsSpace().add(landscape);
        bulletAppState.getPhysicsSpace().add(player);

        BitmapFont guiFont = assetManager.loadFont("Interface/Fonts/Default.fnt");
        infoMovimiento = new BitmapText(guiFont, false);

//infoMovimiento.setSize(guiFont.getCharSet().getRenderedSize());//Tamañoletra
        infoMovimiento.setSize(30);
        infoMovimiento.setColor(ColorRGBA.Red);
        guiNode.attachChild(infoMovimiento);

        infoMovimiento.setLocalTranslation(10, cam.getHeight() - 10, 0);
        infoMovimiento.setText("Controles de navegacion\n"
```

```
            + "Presiona ( W ): Para ir hacia adelante\n"
            + "Presiona ( A ): Para moverte a la izquierda\n"
            + "Presiona ( D ): Para moverte a la derecha\n"
            + "Presiona ( S ): Para ir hacia atras\n"
            + "Presiona ( Espacio ): Para saltar\n"
            + "Usa el raton para mover la camara");
    }
```

Solo colocamos la ruta de nuestro modelo y seria todo:

casona =

assetManager.loadModel("Models/MonteAlbanModel/MonteAlbanModel.j3o");

Resultados


jMonkeyEngine 3.6.1-stable
Controles de navegacion
Presiona (W): Para ir hacia adelante
Presiona (A): Para moverte a la izquierda
Presiona (D): Para moverte a la derecha
Presiona (S): Para ir hacia atras
Presiona (Espacio): Para saltar
Usa el raton para mover la camara


jMonkeyEngine 3.6.1-stable
Controles de navegacion
Presiona (W): Para ir hacia adelante
Presiona (A): Para moverte a la izquierda
Presiona (D): Para moverte a la derecha
Presiona (S): Para ir hacia atras
Presiona (Espacio): Para saltar
Usa el raton para mover la camara

Conclusiones

En conclusión, la graficación juega un papel fundamental e irremplazable en el contexto de la programación y el diseño. Esta disciplina nos proporciona la capacidad de representar información visual de manera efectiva, lo que resulta crucial para una amplia gama de aplicaciones, desde la creación de videojuegos y simulaciones hasta el diseño de experiencias de realidad virtual.

Una de las principales razones por las cuales la graficación es tan importante es su capacidad para comunicar ideas y conceptos de manera visual. A través de gráficos y representaciones visuales, podemos transmitir información compleja de forma más comprensible y accesible para los usuarios, lo que mejora significativamente la experiencia del usuario y facilita la interacción con los sistemas desarrollados.

Además, la graficación desempeña un papel esencial en la creación de mundos virtuales y entornos de simulación. Mediante técnicas avanzadas de renderizado y animación, podemos dar vida a escenarios imaginarios, lo que es especialmente relevante en el desarrollo de videojuegos y aplicaciones de realidad virtual. Esto proporciona una experiencia inmersiva y cautivadora para los usuarios, que pueden explorar y participar en mundos virtuales fascinantes.