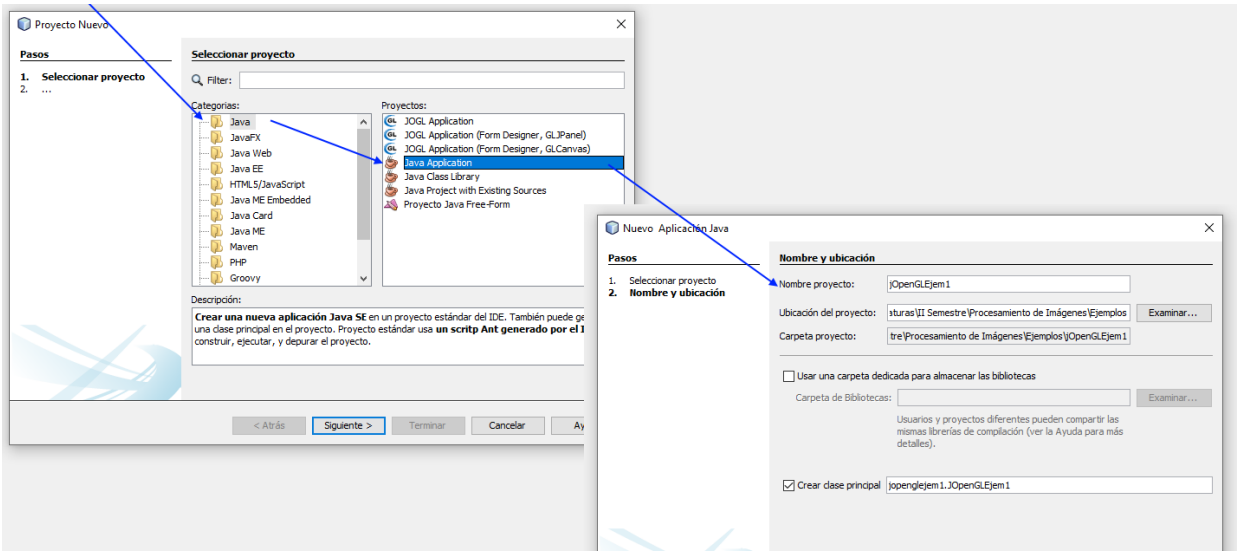
Taller sobre manejo de OpenGL con Java

Para comenzar este taller se debe tener presente los siguientes requisitos; tener instalado:

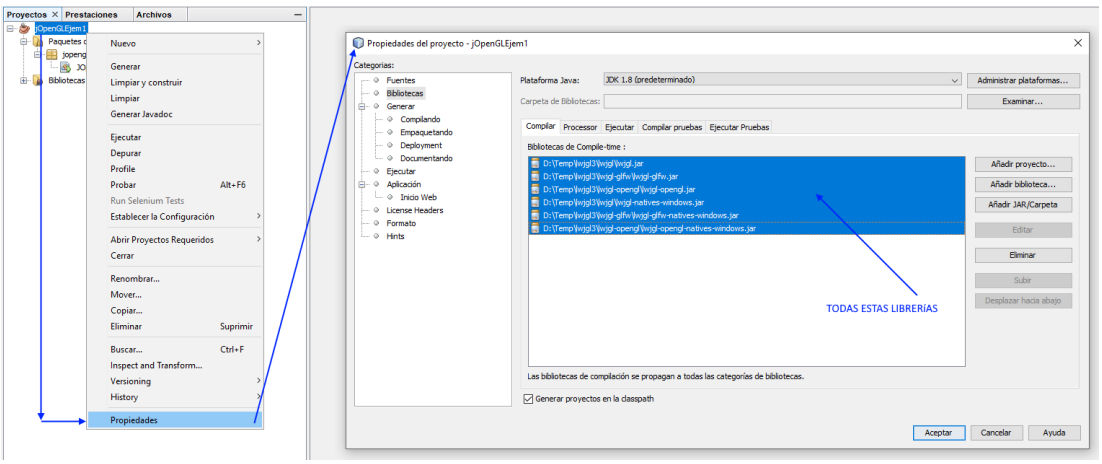
- Instalación realizada en el sistema operativo Windows 10 (64bits).
- Instalar el JDK (usando la versión 1.8.0_65 en adelante): https://www.oracle.com/java/technologies/javase-downloads.html
- NetBeans IDE 8 (Usando la versión 8.2):
- https://netbeans.apache.org/download/index.html
- Descargar el paquete de librerías LWJGL 3 para openGL (desde Github): https://www.lwjgl.org/download (Usando la versión 3.2.3); deberá descargarse el empaquetado en una ubicación temporal (Por ejemplo: D:\Temp\lwjgl3\)

- Configurando un proyecto que ejecute una Ventana con OpenGL (Ejemplo 01):

  - Desde Apache Netbeans, se creará un proyecto:



  - Se añadirán las librerías de "LWJGL" desde la carpeta temporal donde se descargó el paquete para la interacción con *openGL*:
    - lwjgl.jar
    - lwjgl-glfw.jar
    - lwjgl-opengl.jar
    - lwjgl-natives-windows.jar
    - lwjgl-glfw-natives-windows.jar
    - lwjgl-opengl-natives-windows.jar



- Programando la clase principal ***main:***
  - Al igual que con los ejemplos de visual Studio, se creará el método inicializador GLFW, solo que en este caso en vez de usar una estructura (C++) se usará una *clase Java* de nombre "vcd":

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package jopenglejem1;

/**
 *
 * @author PC-MAX
 */
public class vcd {
    private String msj;
    private long ventana;

    public vcd(){
        msj = "";
        ventana = 0;
    }

    public void setMsj(String m){
        msj = m;
    }

    public String getMsj(){
        return msj;
    }

    public void setVentana(long v){
        ventana = v;
    }

    public long getVentana(){
        return ventana;
    }
}
```

  - El evento ***ini_GLFW*** tiene el siguiente código:

```
private static vcd ini_GLFW(long ventana, int ancho, int alto, String titulo){
    vcd objVCD = new vcd();

    glfwInit();
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
    glfwWindowHint(GLFW_OPENGL_PROFILE,GLFW_OPENGL_CORE_PROFILE);
    glfwWindowHint(GLFW_RESIZABLE, GLFW_TRUE); //la ventana podrá ser redimensionada

    //Se construye la ventana  para GLFW
    ventana = glfwCreateWindow(ancho, alto, titulo, NULL, NULL);
    if(ventana == NULL){
        objVCD.setMsj("Fallo al crear la ventana para GLFW");
        objVCD.setVentana(NULL);
        return objVCD;
    }

    //Se configura la redimensión de la ventana usando a glViweport()
    glfwSetFramebufferSizeCallback(ventana, (objVentana, an, al) -> {
        glViewport(0, 0, an, al);
    });

    //Se obtiene la resolución del monitor principal para mostrar la ventana en forma centrada
    GLFWVidMode vidmode = glfwGetVideoMode(glfwGetPrimaryMonitor());
    glfwSetWindowPos(ventana,(vidmode.width() - ancho) / 2,(vidmode.height() - alto) / 2); //Centrar ventana

    glfwMakeContextCurrent(ventana); //Colocar la ventana como hilo de proceso actual
    glfwSwapInterval(1); // Activar la sincronización vertical en el proceso actual(1), para evitar el parpadeo entre búfers
    objVCD.setVentana(ventana); //Asignar la ventana creada al objeto objVCD para su retorno (en este caso, sin obs)
    return objVCD;
}
```

glfwCreateWindow se encarga de crear la ventana y enviar la alerta en caso haber fracasado, esto se captura mediante el objeto ***objVCD***.

Algo muy importante aquí, son los métodos ***Callback;*** pues que, no hay necesidad de crear una función como en VS. ya que al definirlas en esta sección (*glfwSetFramebufferSizeCallback, glfwSetKeyCallback*) el programa sabrá desplazarse en estas zonas, para realizar su objetivo. (las funciones de los Callback ya fueron vistas en problemas anteriores y permanecen de forma similar.

Un evento optativo es la redimensión de la ventana y ubicarla en forma concéntrica, esto se logra con *GLFWVidMode* y se posiciona con *glfwSetWindowPos*

Otro evento a tener en cuenta es *glfwSwapInterval* ya que activa la sincronización vertical en el proceso actual → 1, para evitar el parpadeo entre búfers.

- Finalmente, el "main" quedaría así:

```
public static void main(String[] args) {
    long ventana = NULL;
    vcd venConDat = new vcd();
    try {
        //Se inicializa GLFW, asimismo si todo es correcto, se crea la ventana
        venConDat = ini_GLFW(ventana,1024, 720, "Ejem 1: OpenGL con Java");
        if (!"".equals(venConDat.getMsj())) {
            System.out.println(venConDat.getMsj()); //imprime el error cometido
            glfwTerminate(); //limpia y termina la aplicación
        }
        else {
            ventana = venConDat.getVentana();
        }
        GL.createCapabilities(); //Linea elemental para hacer interactuar el contexto openGL con GLFW
        glClearColor(1.0f, 0.0f, 0.0f, 0.0f); //Configurar color de fondo

        while (!glfwWindowShouldClose(ventana)) {
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // limpiar el FrameBuffer
            glfwSwapBuffers(ventana); //activar el doble búfer
            glfwPollEvents(); //capturar cualquier interacción del usuario con la ventana
        }
        glfwFreeCallbacks(ventana); //Liberar las llamadas que interactuen con la ventana
        glfwDestroyWindow(ventana); //Liberar memoria de la ventana creada
    } finally {
        glfwTerminate(); //limpia y termina la aplicación
    }
}
```

Como se aprecia, se crea el objeto *venConDat* y se controla con una condición en caso hubiera errores.

El evento *GL.createCapabilities()* de la librería OpenGL, se encarga de vincular el contexto actual con GLFW. Asimismo, se define el color de fondo *glClearColor* (evento ya explicado en ejercicios anteriores)

Una vez que se termine de ejecutar el ciclo (bucle) al cerrarse la ventana o indicar su cerrado, debe eliminarse el compromiso con la memoria asignada por los Callback de la ventana actual (*glfwFreeCallbacks*) y posteriormente destruir dicha ventana. Es necesario también ejecutar *glfwTerminate()* para liberar cualquier otro objeto usado con las librerías.

- El programa se cargar de mostrar una ventana de color de fondo rojo:

- Material de estudio de donde se desarrollaron los pasos anteriores:
  - http://acodigo.blogspot.com/2016/12/opengl-en-java-lwjgl.html
  - https://www.youtube.com/watch?v=ZBWAXr9z0bI