6. Texturas

Descarga del campus las imágenes (archivos .bmp de BmpsP1) y Texturas.zip. Copia los archivos .bmp en el directorio Bmps del proyecto y los archivos de código (.h y .cpp) en el directorio IG1App. Y añade al proyecto los nuevos archivos de código (Explorador de soluciones -> Agregar -> Elemento existente).

Modifica la clase Mesh para incorporar el vector de coordenadas de textura: añade un nuevo atributo (vector<dvec2> vTexCoords) y adapta el método render().

En la clase Scene, añade un atributo para las texturas de los objetos (vector<Texture*> gTextures). En el método init, crea y carga las texturas (con el método load de Texture) para los objetos de la escena. Adapta los métodos free, setGL (activa las texturas en OpenGL), y resetGL (las desactiva).

Añade a las entidades un atributo Texture* mTexture = nullptr y un método setTexture(Texture* tex) { mTexture = tex; }. En el método render de las entidades, activa (método bind de Texture) la textura antes de renderizar la malla, y desactívala después (unbind).

7. Cambio de escena

Para cambiar entre escenas, añade a la clase Scene un atributo para el identificador de escena int mId = 0 y un método para cambiarlo void setState(int id) (si es necesario, reinicia). Con la tecla 1 cambiamos a la escena 1 (escena 3D) y con la tecla 0 a la 0 (escena 2D). Modifica el método init para que, en función de mId inicie una escena u otra. Tendrás que adaptar otros métodos (free, resetGL, initGL, …).

8. Animación (Opcional)

Añade, a la clase IG1App, el método update() (sin argumentos) y la función estática s_update() (que invoca al método update()) para el callback de glutIdleFunc. Está función será llamada cuando la aplicación esté desocupada y la utilizamos para actualizar los valores de animación. El método update() debe indicar a la escena que se actualice cada cierto tiempo (no más de 60 veces por segundo). Para esto, añade una variable (GLuint mLastUpdateTime) para capturar el último instante en que se realizó una actualización y utiliza glutGet(GLUT_ELAPSED_TIME) (devuelve los milisegundos transcurridos desde que se inició) para actualizar la variable y controlar el tiempo que debe transcurrir entre actualizaciones. Añade también una variable bool para activar/desactivar la animación con la tecla U.

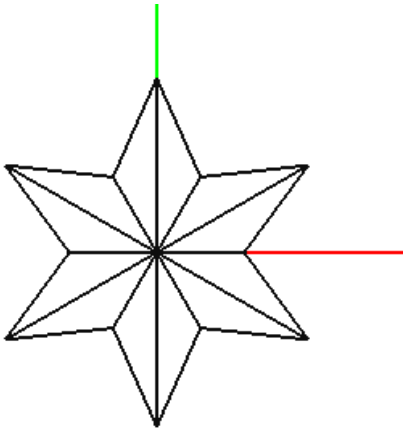
9. Estrella 3D

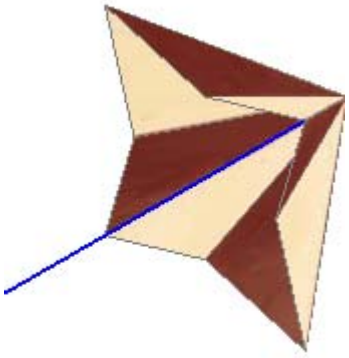
Define la función static Mesh* generaEstrella3D(GLdouble re, GLuint np, GLdouble h) que genera los vértices de una estrella de np puntas, centrada en el plano Z=h. Utiliza la primitiva GL_TRIANGLE_FAN con primer vértice V0 = (0, 0, 0).

El número de vértices es 2*np + 2.

Utiliza la ecuación de la circunferencia para generar los vértices. Puedes añadir un parámetro ri para el radio interior o utilizar re/2.

Recuerda generar los vértices en orden contrario a las agujas del reloj (CCW).





Define la clase Estrella3D heredando de Abs_Entity, y añade una entidad de esta clase a la escena (renderiza en modo líneas).

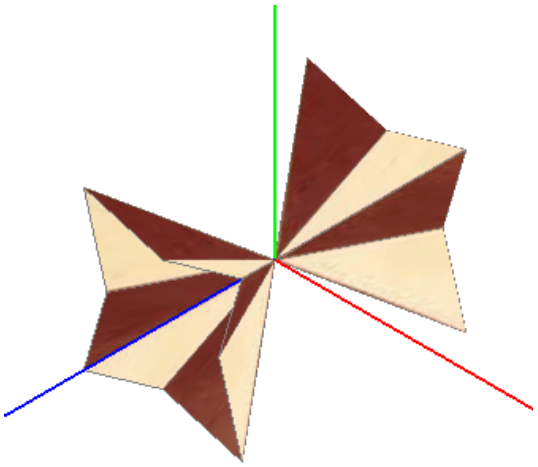
Modifica el método render de la clase Estrella3D para dibujar dos veces la estrella según aparece en la imagen del punto 10. La misma malla se renderiza dos veces con distinta matriz de modelado.

Animación: Añade a la clase Estrella3D atributos para que gire sobre su eje Z y sobre su eje Y (los ángulos de giro). Redefine el método update para actualizar los ángulos y la matriz mModelMat.

10. Estrella con textura

Define la función static Mesh* generaEstrellaTexCor(GLdouble re, GLuint np, GLdouble h) que añade coordenadas de textura a la estrella centrando la imagen en su vértice (0,0,0).

Modifica la constructora de Estrella3D y el método render para renderizar la estrella con textura.



11. Suelo con textura

Añade la clase Suelo: Entidad que renderiza un rectángulo centrado en el plano Y=0, embaldosado con una textura que se repite. En la constructora, utiliza la malla generaRectangulo() y establece la matriz de modelado para posicionarla horizontal.

Define la función static Mesh* generaRectanguloTexCor(GLdouble w, GLdouble h, GLuint rw, GLuint rh) que añade coordenadas de textura para cubrir el rectángulo con una imagen que se repite rw veces a lo ancho y rh a lo alto.

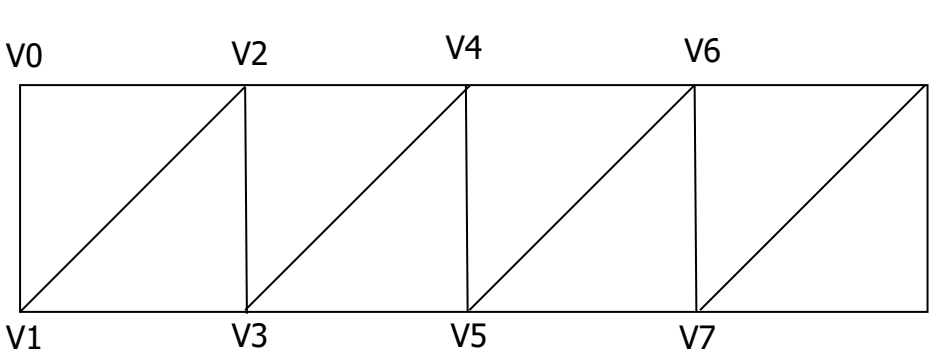
Modifica la constructora de Suelo y el método render para renderizar el suelo con textura.

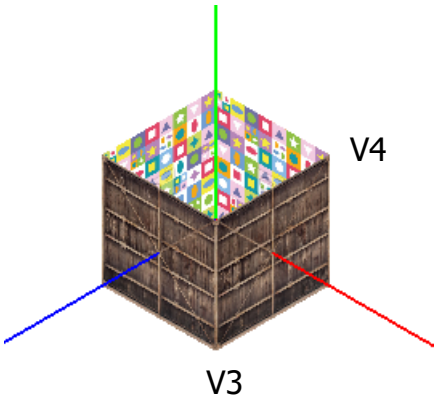
Ajusta el número de repeticiones de la textura según las dimensiones del rectángulo y de la textura, de forma que la imagen guarde sus proporciones y no pierda calidad.

Establece un color para modularlo con la textura.

12. Caja

Define la función static Mesh* generaContCubo(GLdouble ld) que genera los vértices del contorno de un cubo, centrado en los tres ejes, de lado ld. Utiliza la primitiva GL_TRIANGLE_STRIP.





El número de vértices es 10: 8 del cubo (V0,…, V7) + 2 para cerrar el contorno (V0, V1).

Define la clase Caja heredando de Abs_Entity, y añade una entidad de esta clase a la escena (renderiza en modo líneas).

13. Fondo de la caja (Opcional)

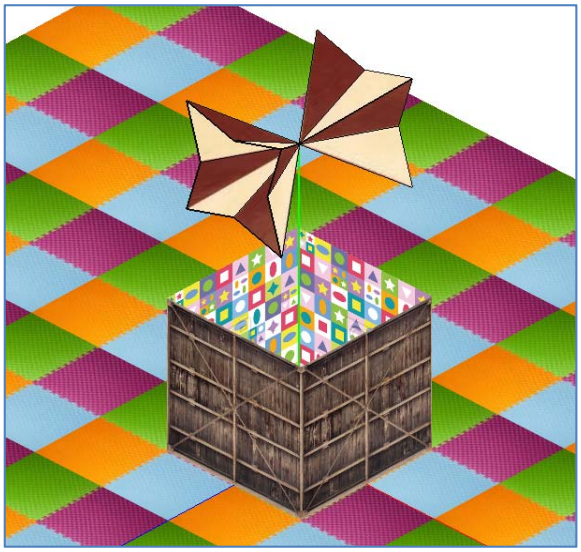
Añade a la clase Caja un rectángulo para el fondo: añade otro atributo Mesh*, y una matriz para colocarla en el fondo.

14. Caja con textura

Define la función static Mesh* generaCajaTexCor(GLdouble nl) que añade coordenadas de textura a la caja, repitiendo la imagen en cada lado.

Modifica el método render para renderizar la caja con dos texturas (añade otro atributo Texture*), una para el exterior de la caja y otra para el interior (incluido el fondo de la caja). Para renderizar solo el exterior o el interior, utiliza los comandos:

glEnable(GL_CULL_FACE)
glCullFace(GL_FRONT / GL_BACK)
glDisable(GL_CULL_FACE)



Escena 3D

Suelo (rectángulo centrado en el plano Y=0), la caja sobre el suelo en el cuadrante –X, -Z, y la estrella por encima de la caja

15. Foto

Nueva entidad que renderiza un rectángulo sobre el suelo con una textura cuya imagen es la visualización de la escena en el renderizado anterior (front buffer). Puedes acceder a las dimensiones de la ventana añadiendo métodos de consulta a IG1App, por ejemplo: IG1App::s_ig1app.winWidth();

Añade a la clase Texture el método loadColorBuffer(…) con los comandos de OpenGL necesarios para copiar el color buffer en la textura: glCopyTexImage2D, glReadBuffer

Define le método update para que se actualice la textura.

16. Guardar en archivo bmp (Opcional)

Define la tecla F para guardar una imagen resultante del renderizado en un archivo bmp.

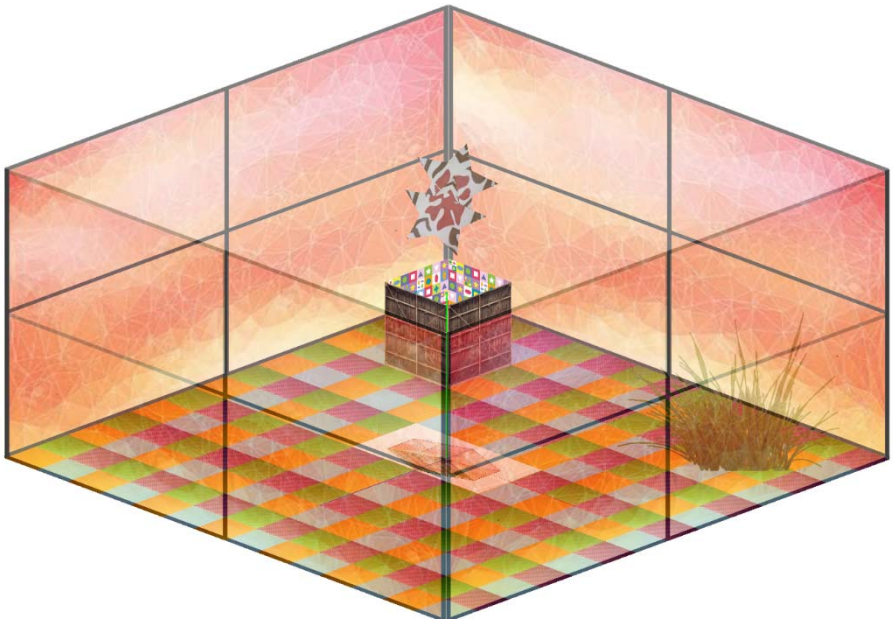
Añade a la clase Texture el método save(const std::string & BMP_Name). Para implementarlo utiliza: una variable local de la clase PixMap32RGBA; el comando glGetTexImage(…) para obtener los datos de la textura en la variable (la variable tiene que tener memoria para la textura); y el método save_bmp24BGR() de la clase PixMap32RGBA para guardar la imagen en el archivo.

17. Cristalera translúcida (Blending)

Contorno de un cubo que aparece alrededor del suelo con una textura con todos los colores translúcidos.

18. Planta (Opcional)

Utiliza la imagen grass.bmp, con el color de fondo transparente, como textura de al menos dos rectángulos que se cruzan en el eje Y, y que aparecen sobre el suelo en una esquina.



19. Escena 3D

Suelo (rectángulo centrado en el plano Y=0), la caja sobre el suelo en el cuadrante –X, -Z, la estrella por encima de la caja, la foto y la cristalera.

Animación de la estrella y actualización de la foto.