



Universidad Nacional Autónoma de México.



Facultad de Ingeniería.

Ingeniería en Computación.

Laboratorio de Computación Gráfica e Interacción Humano Computadora.

Alumnos:

- Brito Segura Ángel.
- Hernández Torres Agustín de Jesús.
- Huarte Nolasco Mario.

Documento: Manual Técnico.

Grupo de Teoría: 03.

Fecha límite de entrega: 25 / julio / 2021.

Semestre: 2021-2.

Antes de ejecutar el programa:

Para llevar a cabo las siguientes instrucciones se considera que el usuario cuenta con el IDE Visual Studio 2017 o posterior, además de contar con los paquetes de OpenGL ya instalados para su implementación.

Creación del proyecto en Visual Studio:

1. Se abre el IDE Visual Studio.

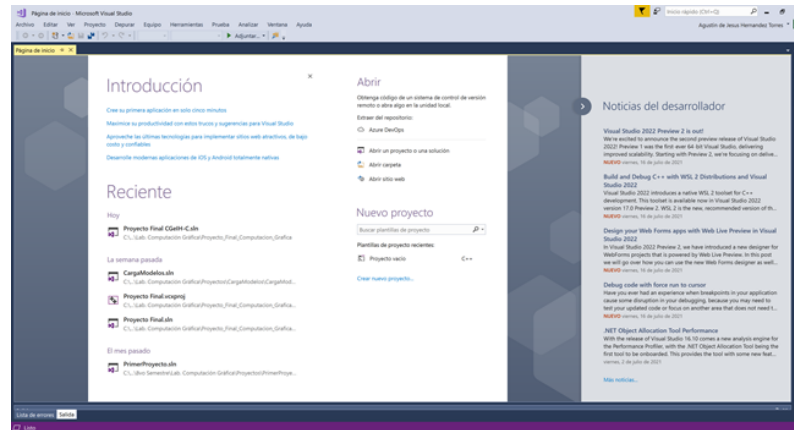


Figura 1: Pantalla inicial de Visual Studio.

2. Se crea un nuevo proyecto, asignando su nombre, ubicación y nombre de la solución (generalmente el mismo nombre que el proyecto).

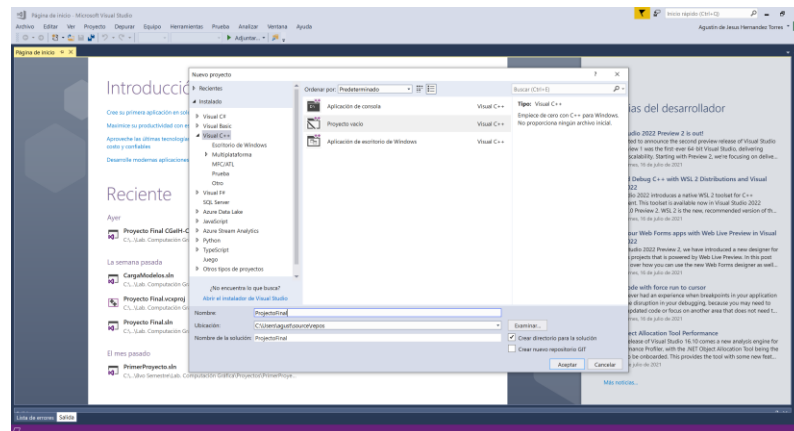


Figura 2: Creación de proyecto en Visual Studio.

3. Ya que se tiene el proyecto creado, se agregan todos los archivos necesarios para el correcto funcionamiento del programa, los cuales incluyen archivos de encabezado (con extensión .h), archivos de origen (con extensión .cpp) y archivos de recursos (con extensión .frag o .vert). Cada tipo de archivo se añade en su respectiva carpeta. Para realizar lo

anterior, se abre el explorador de soluciones (parte superior derecha de la pantalla) y se da clic derecho en cada carpeta, seleccionando “Agregar” y posteriormente “Elemento existente”.

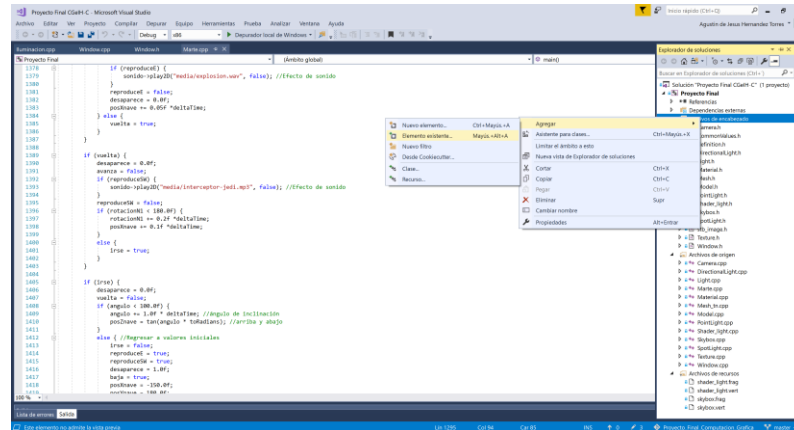


Figura 3: Añadido de archivos en su carpeta correspondiente.

4. Al terminar de agregar todos los archivos, se realizan configuraciones en el proyecto. Para esto, se da clic derecho en el nombre del proyecto y posteriormente en “Propiedades”

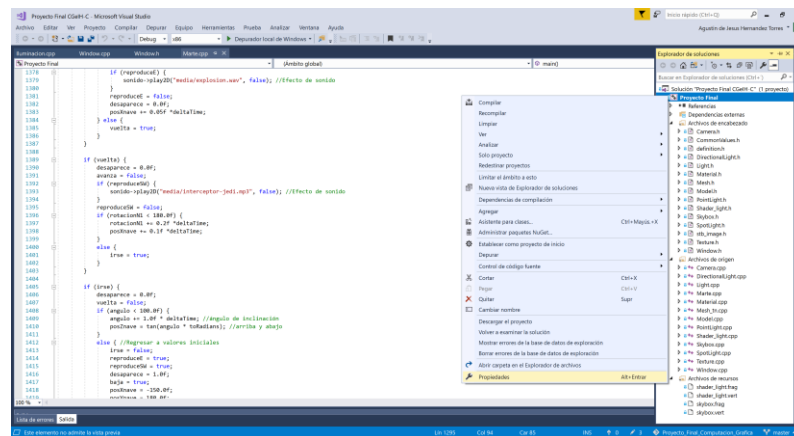


Figura 4: Añadido de archivos en su carpeta correspondiente.

5. Al hacer lo anterior, se despliega una ventana de configuraciones. En dicha ventana se agregan los archivos de la carpeta “lib”, pues son esenciales para que el proyecto se ejecute. Esto se realiza seleccionando el “Vinculador”, posteriormente, en la sección de “General” y finalmente en la parte de “Directorios de bibliotecas adicionales” se agrega “lib;”. Además, en la parte de “Vincular dependencias de bibliotecas” se escribe “Si”.

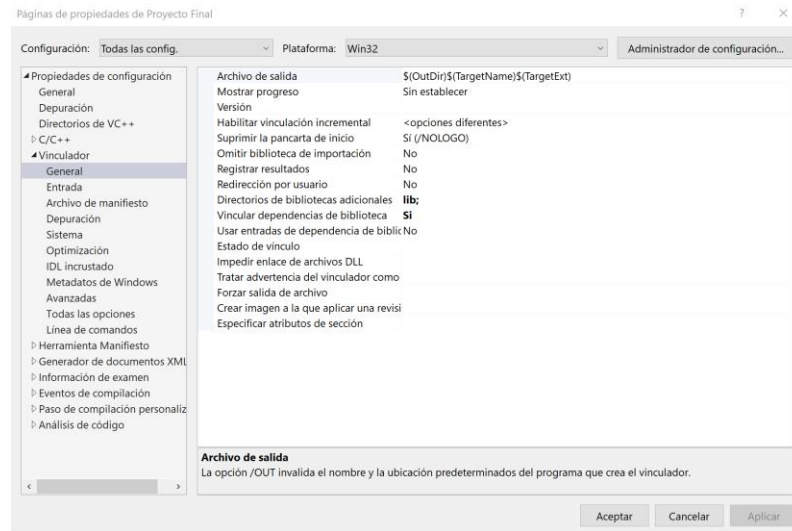


Figura 5: Configuración en el Vinculador/General.

6. De igual manera, en la misma ventana se va a la sección de “C/C++”, posteriormente, en la sección de “General” y finalmente en la parte de “Directorios de inclusión adicionales”. En dicha parte se agregan “include;glm;”.

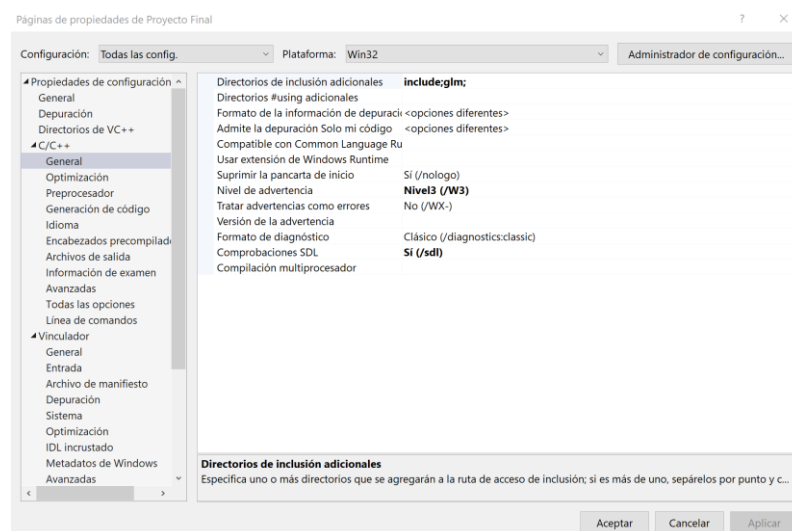


Figura 6: Configuración en C/C++/General.

7. Finalmente, de nuevo en la sección de “Vinculador”, en la parte de “Dependencias adicionales” se agrega la siguiente línea: “assimp-vc140-mt.lib;glew32.lib;glfw3.lib;opengl32.lib;”. Estos archivos son dependencias que permiten ejecutar y visualizar de forma correcta el resultado del proyecto.

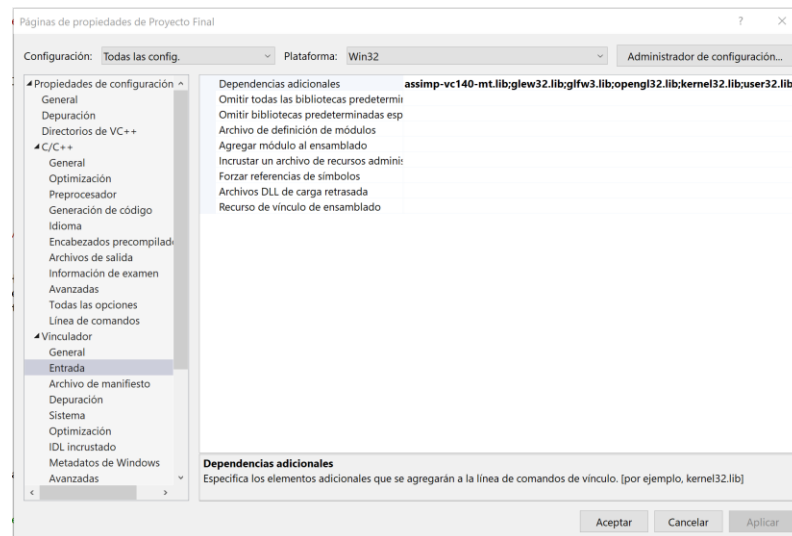


Figura 7: Configuración en Vinculador/Entrada.

8. Con las configuraciones anteriores, el proyecto está listo para ser ejecutado desde el Depurador local.

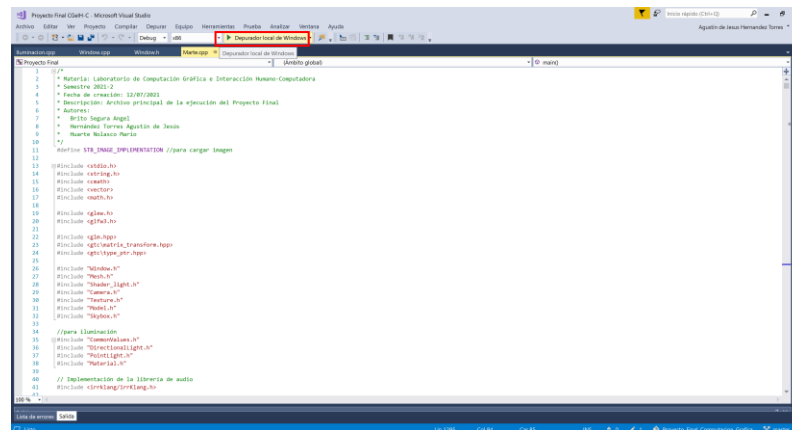


Figura 8: Proyecto en Visual Studio listo para compilarse y ejecutarse.

Bloques de código que permiten la creación de elementos dentro del escenario.

Para crear todo el escenario se definen modelos, texturas y skybox para la implementación de estos dentro del proyecto. Esto se logra declarando los

Se observa que todos los elementos a incluir en el escenario se encuentran definidos en dicha sección. Posteriormente, se cargan las texturas, modelos e imágenes que harán posible la implementación del skybox. Para todos los elementos se indica la ruta desde la cual va a tomar cada textura, objeto o imagen, dependiendo de lo que se vaya a cargar. En la siguiente imagen se muestra cómo se hace la declaración para las texturas, los modelos y el skybox de día y de noche respectivamente.

[illegible]

Figura 10: Carga de modelos, texturas y skybox.

Además, se realiza la implementación de luces de tipo “Spot Light” y “Point Light”, las cuales se realizan de la siguiente manera.

```

100 //Contador de luces puntuales (no se requiere si solo se tiene una sola luz)
101 unsigned int pointLightCount = 0;
102
103 //Declaración de primer luz puntual
104 pointLight[0] = PointLight(0.0f, 0.0f, 0.0f, //sin color
105 1.0f, 1.0f, //coeficientes
106 2.0f, 1.0f, 1.0f, //posición dentro del escenario
107 0.0f, 0.0f, 0.0f) //sección de segundo grado para la atenuación = 1u + c
108 pointLightCount++;
109
110 pointLight[1] = PointLight(1.0f, 1.0f, 1.0f, //color blanco
111 1.0f, 1.0f, //coeficientes
112 -100.0f, 1.0f, 20.0f, //posición dentro del escenario
113 0.0f, 0.0f, 0.0f) //sección de segundo grado para la atenuación = 1u + c
114 pointLightCount++;
115
116 pointLight[2] = PointLight(1.0f, 1.0f, 1.0f, //color blanco
117 1.0f, 1.0f, //coeficientes
118 -100.0f, 1.0f, 20.0f, //posición dentro del escenario
119 0.0f, 0.0f, 0.0f) //sección de segundo grado para la atenuación = 1u + c
120 pointLightCount++;
121
122 pointLight[3] = PointLight(1.0f, 1.0f, 1.0f, //color blanco
123 1.0f, 1.0f, //coeficientes
124 -100.0f, 1.0f, 20.0f, //posición dentro del escenario
125 0.0f, 0.0f, 0.0f) //sección de segundo grado para la atenuación = 1u + c
126 pointLightCount++;
127
128 pointLight[4] = PointLight(1.0f, 1.0f, 1.0f, //color blanco
129 1.0f, 1.0f, //coeficientes
130 -100.0f, 1.0f, 20.0f, //posición dentro del escenario
131 0.0f, 0.0f, 0.0f) //sección de segundo grado para la atenuación = 1u + c
132 pointLightCount++;
133
134 pointLight[5] = PointLight(1.0f, 1.0f, 1.0f, //color blanco
135 1.0f, 1.0f, //coeficientes
136 -100.0f, 1.0f, 20.0f, //posición dentro del escenario
137 0.0f, 0.0f, 0.0f) //sección de segundo grado para la atenuación = 1u + c
138 pointLightCount++;

```

Figura 11: Declaración e implementación de algunas “Point Lights”.

```

100 unsigned int spotLightCount = 0;
101
102 //luz de tipo spot
103 spotLight[0] = SpotLight(1.0f, 1.0f, 1.0f, //color blanco
104 0.0f, 2.0f, //coeficientes
105 0.0f, 0.0f, 0.0f, //posición dentro del escenario
106 0.0f, 1.0f, 0.0f, //sección de segundo grado para la atenuación = 1u + c
107 2.0f, 0.0f, 0.0f, //radio del cono
108 1.0f) //apertura (radio) del cono entre más grande será más grande la circunferencia
109 spotLightCount++;
110
111 //luz de tipo spot
112 spotLight[1] = SpotLight(1.0f, 1.0f, 1.0f, //color blanco
113 1.0f, 1.0f, //coeficientes
114 -1.0f, 0.0f, 0.0f, //posición dentro del escenario
115 0.0f, 0.0f, 0.0f, //sección de segundo grado para la atenuación = 1u + c
116 0.0f, 0.0f, 0.0f, //radio del cono
117 10.0f) //apertura (radio) del cono entre más grande será más grande la circunferencia
118 spotLightCount++;
119
120 //luz de tipo spot
121 spotLight[2] = SpotLight(1.0f, 0.0f, 0.0f, //color rojo
122 1.0f, 1.0f, //coeficientes
123 -100.0f, 0.0f, 0.0f, //posición dentro del escenario
124 0.0f, 0.0f, 0.0f, //sección de segundo grado para la atenuación = 1u + c
125 1.0f, 0.0f, 0.0f, //radio del cono
126 10.0f) //apertura (radio) del cono entre más grande será más grande la circunferencia
127 spotLightCount++;
128
129 spotLight[3] = SpotLight(0.0f, 0.0f, 1.0f, //color verde
130 1.0f, 1.0f, //coeficientes
131 -100.0f, 0.0f, 0.0f, //posición dentro del escenario
132 0.0f, 0.0f, 0.0f, //sección de segundo grado para la atenuación = 1u + c
133 1.0f, 0.0f, 0.0f, //radio del cono
134 10.0f) //apertura (radio) del cono entre más grande será más grande la circunferencia
135 spotLightCount++;

```

Figura 11: Declaración e implementación de algunas “Spot Lights”.

Finalmente, se declara la información espacial de los objetos o modelos que se cargan en el proyecto para que aparezcan en la escena. Dicha información es en dónde va a aparecer (traslación), en qué posición va a aparecer (rotación) y de qué tamaño va a aparecer (escala). Además, si son modelos con jerarquía se debe declarar otra matriz auxiliar. Un ejemplo de implementación es el siguiente:

```

100 // Matriz de transformación
101 model = glm::mat4(1.0f);
102 model = glm::translate(model, glm::vec3(0.0f, 1.0f, 0.0f));
103 model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
104 glBindVertexArray(vao);
105 glDrawArrays(GL_TRIANGLES, 0, 36);
106 glBindBuffer(GL_ARRAY_BUFFER, 0);
107
108 // Matriz de transformación
109 model = glm::mat4(1.0f);
110 model = glm::translate(model, glm::vec3(0.0f, 1.0f, 0.0f));
111 model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
112 model = glm::rotate(model, 90 * glm::radians(1.0f), glm::vec3(1.0f, 0.0f, 0.0f));
113 glBindVertexArray(vao);
114 glDrawArrays(GL_TRIANGLES, 0, 36);
115 glBindBuffer(GL_ARRAY_BUFFER, 0);
116
117 // Matriz de transformación
118 model = glm::mat4(1.0f);
119 model = glm::translate(model, glm::vec3(0.0f, 1.0f, 0.0f));
120 model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
121 model = glm::rotate(model, 90 * glm::radians(1.0f), glm::vec3(1.0f, 0.0f, 0.0f));
122 glBindVertexArray(vao);
123 glDrawArrays(GL_TRIANGLES, 0, 36);
124 glBindBuffer(GL_ARRAY_BUFFER, 0);
125
126 // Matriz de transformación
127 model = glm::mat4(1.0f);
128 model = glm::translate(model, glm::vec3(0.0f, 1.0f, 0.0f));
129 model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
130 model = glm::rotate(model, 90 * glm::radians(1.0f), glm::vec3(1.0f, 0.0f, 0.0f));
131 glBindVertexArray(vao);
132 glDrawArrays(GL_TRIANGLES, 0, 36);
133 glBindBuffer(GL_ARRAY_BUFFER, 0);
134
135 // Matriz de transformación
136 model = glm::mat4(1.0f);
137 model = glm::translate(model, glm::vec3(0.0f, 1.0f, 0.0f));
138 model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
139 model = glm::rotate(model, 90 * glm::radians(1.0f), glm::vec3(1.0f, 0.0f, 0.0f));
140 glBindVertexArray(vao);
141 glDrawArrays(GL_TRIANGLES, 0, 36);
142 glBindBuffer(GL_ARRAY_BUFFER, 0);

```

Figura 11: Declaración e implementación de algunos modelos u objetos.

Finalmente, se dejan los enlaces de donde se obtuvieron algunos modelos, librerías de audio y audios utilizados:

Modelos tomados de Internet:

- Star Fighter:

<https://sketchfab.com/3d-models/jedi-star-fighter-0b641c2f2b854f1f9ae7f2a731e44dbd>

- Speeder Bike:

<https://sketchfab.com/3d-models/star-wars-barc-speeder-bike-56ff0b8c18744664aa121db5d2039eb3>

- Stormtrooper:

<https://sketchfab.com/3d-models/dikwrnkrt9ts-stormtrooper-ecf2ebbb6880485f936047929b33fe57>

- Wall-E:

<https://sketchfab.com/3d-models/wall-e-modeling-c64a10b13e2a45efb98f9780553a2140>

- EVA:

<https://sketchfab.com/3d-models/wall-e-eve-91ba6b0cdb9d45da9cb7c014d348aff3>

- Finn y Jake:

<https://sketchfab.com/3d-models/finn-and-jake-4d739e8a98234e70b89e274ded543b28>

- Estrella de la Muerte:

<https://sketchfab.com/3d-models/death-star-star-wars-699e5f635f7a4b21a4c0d2e3f5b68324>

- Estrellas:

<https://sketchfab.com/3d-models/space-vehicle-4b9b78c0a72b45fe9ea9b3920b49c906>

Imagen base para Skybox:

<https://www.pngegg.com/en/png-oeswi>

Librería de audio utilizada:

<https://learnopengl.com/In-Practice/2D-Game/Audio>

<https://www.ambiera.com/irrklang/tutorials.html>

<https://www.ambiera.com/irrklang/downloads.html>

Audios introducidos:

Soundtrack: <https://youtu.be/UmccvCkflpU>

Interceptor-Jedi: <https://youtu.be/e9lapdvLSGw>

Wall-E y Eva: <http://www.moviesoundclips.net/sound.php?id=158>

<http://www.sonidosmp3gratis.com/download.php?id=15430&sonido=wall%20e%20eva>

Speeder

Bike:

<http://www.sonidosmp3gratis.com/download.php?id=16180&sonido=star%20wars%20dogfight>

Yoda: <http://www.sonidosmp3gratis.com/download.php?id=15673&sonido=yoda>

Darth Vader: http://www.tonosfrikis.com/buscar/melodias/star_wars