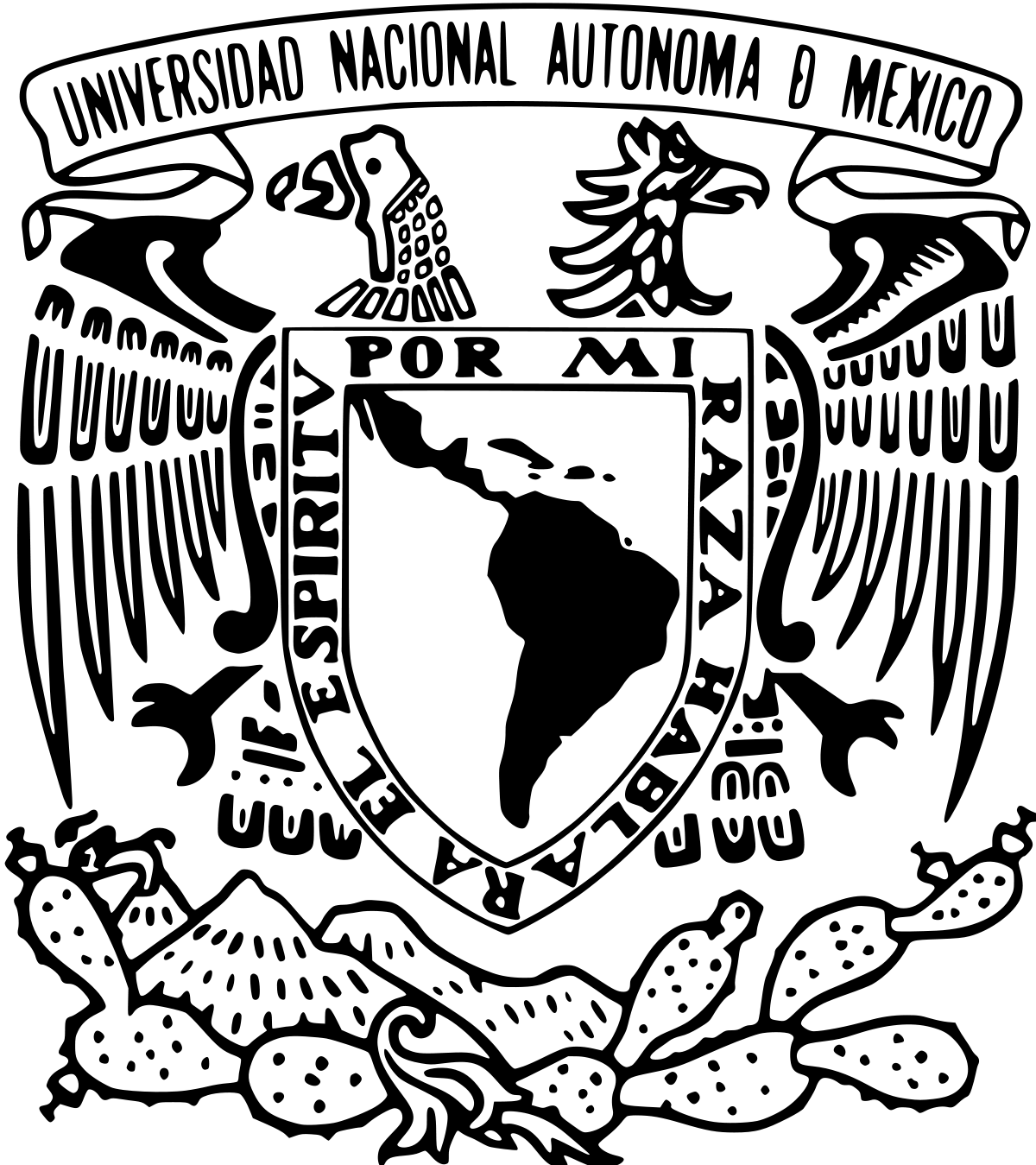
**Universidad Nacional Autónoma De México**

***Facultad de Ingeniería***

*Computación Gráfica e Interacción Humano-Computadora*



**Prof.: Carlos Aldair Roman Balbuena**

***Proyecto Final***

**Documentación Proyecto Final**

**Carranza Escobar Luis Enrique Enrique**

***Grupo: 04, Semestre: 2021-2***

*Fecha de Entrega: martes, 10 de agosto de 2021*



Índice

[Objetivos 3](#_Toc79263479)

[Alcance 3](#_Toc79263480)

[Requerimientos 3](#_Toc79263481)

[Cronograma de actividades 4](#_Toc79263482)

[Especificaciones de hardware y software 4](#_Toc79263483)

[Hardware 4](#_Toc79263484)

[Software 5](#_Toc79263485)

[Herramientas 5](#_Toc79263486)

[Estimación de costos 6](#_Toc79263487)

[Desarrollo 7](#_Toc79263488)

[Modelos 9](#_Toc79263489)

[Modelos animados 15](#_Toc79263490)

[Casos de uso animaciones 16](#_Toc79263491)

[Diccionario de datos 17](#_Toc79263492)

[Diccionario de funciones 22](#_Toc79263493)

[Proyecto en Github 23](#_Toc79263494)

[Referencias Modelos 23](#_Toc79263495)

# Objetivos

* Guiar y brindar las herramientas necesarias para aquella persona que desee continuar con el proyecto, ya sea para mejorarlo o para base de otro proyecto.
* Mostrar la configuración del sistema y las especificaciones mínimas de hardware y software.

# Alcance

* Explicar el funcionamiento y el control del proyecto con el uso de Xcode y el lenguaje de programación C++.
* Explicar la configuración del entorno de desarrollo integrado.

# Requerimientos

Recrear una unidad habitacional con edificios, departamentos, autos, personas dentro del entorno. Todo esto deberá verse en 3D. Uno de los departamentos de los edificios debe estar ambientado con al menos 7 modelos y 4 animaciones que vayan acorde a lo que se diseñó.

La siguiente imagen servirá como imagen de referencia:



Los edificios de color blanco van a ir cambiando por otros edificios departamentales.

# Cronograma de actividades

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre actividad | Fecha Inicio | Duración en días | Fecha Fin |
| Planificación de actividades | 22-jun | 1 | 23-jun |
| Estimación de costos | 22-jun | 1 | 23-jun |
| Presentación Propuesta | 24-jun | 1 | 25-jun |
| Preparación de Software | 23-jun | 3 | 26-jun |
| Búsqueda de recursos | 27-jun | 6 | 04-jul |
| Prueba de recursos | 03-jul | 3 | 06-jul |
| Generación del entorno | 10-jul | 1 | 11-jul |
| Generación de los objetos | 11-jul | 1 | 12-jul |
| Programación de animaciones | 13-jul | 4 | 17-jul |
| Asignación de eventos | 18-jul | 4 | 22-jul |
| Generación de la documentación | 27-jul | 6 | 06-ago |
| Generación del video | 07-ago | 1 | 07-ago |
| Entrega proyecto final | 09-ago | 1 | 10-ago |

# Especificaciones de hardware y software

## Hardware

* Procesador Intel Core i5 de dos núcleos
* 8 GB de memoria RAM DDR3
* Intel Iris Graphics 6100 1536 MB

## Software

* macOS Big Sur versión 11.4
* Xcode Versión 12.4 (12D4e)
* Assimp versión 5
* SDL2
* GLFW versión 3.3
* GLEW versión 2.2
* OpenGL: framework

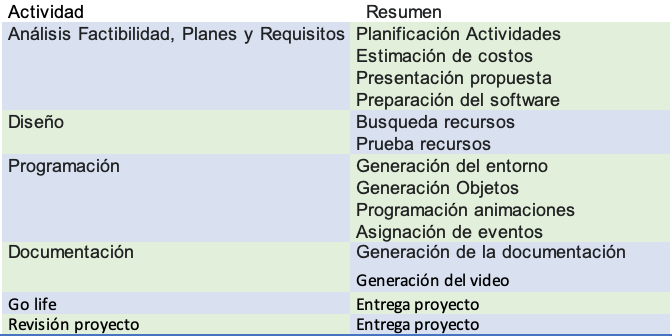
# Herramientas

|  |  |
| --- | --- |
| **Herramienta** | **Descripción** |
| Blender | Software­­ dedicado a la edición de modelos en 3D, generación de archivos OBJ y MTL que son los necesarios para esté proyecto |
| Xcode | IDE de programación de la compañía Apple |
| C++ | Lenguaje de programación con el que se llevará acabo el desarrollo del proyecto |
| GIMP-2.10 | Software encargado de la edición de imágenes |
| 3D MAX | Software dedicado a la edición de modelos en 3D, generación de archivos OBJ y MTL que son los necesarios para esté proyecto |
| CGTRADER | Página web donde buscaremos los modelos necesarios |
| GITHUB | Plataforma online donde alojaremos todo el proyecto |

# Estimación de costos

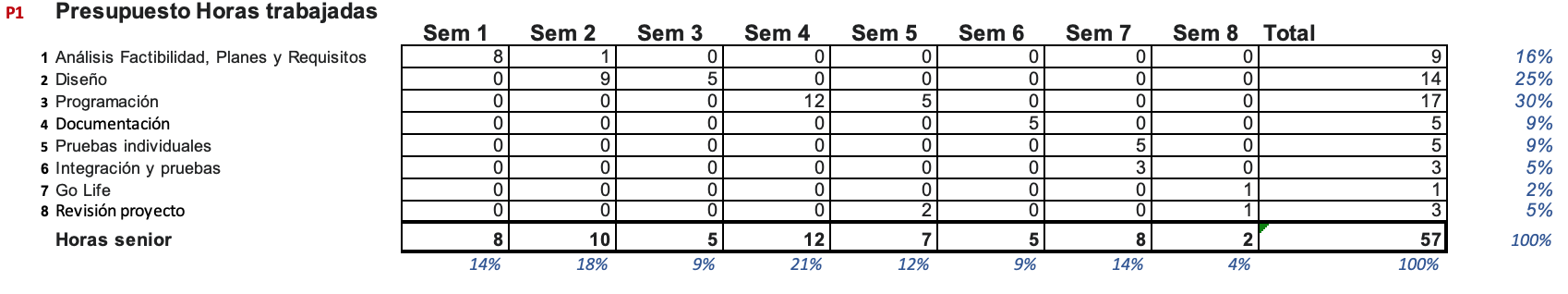
Primero, para obtener nuestro presupuesto fue que todas las actividades las englobamos en 8 actividades principales que son Análisis Factibilidad Planes y Requisitos, Diseño Programación, Documentación, Pruebas individuales, Integración y pruebas, Go Life y Revisión proyecto.

La agrupación queda de la siguiente manera:

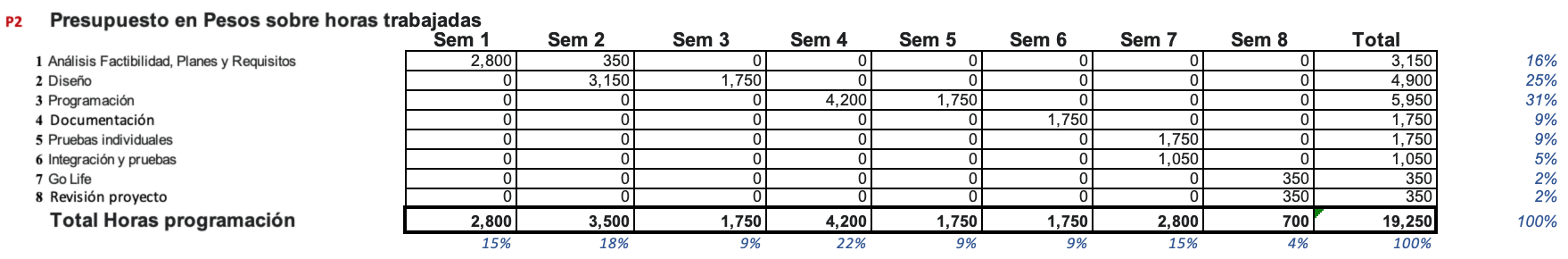


Una vez realizada la agrupación lo que hice fue hacer una tabla que iba a relacionar la actividad hecha por el tiempo en horas que me llevaría hacer la tarea, por lo tanto, tenemos:

Obtuvimos un aproximado en horas trabajadas en 8 semanas



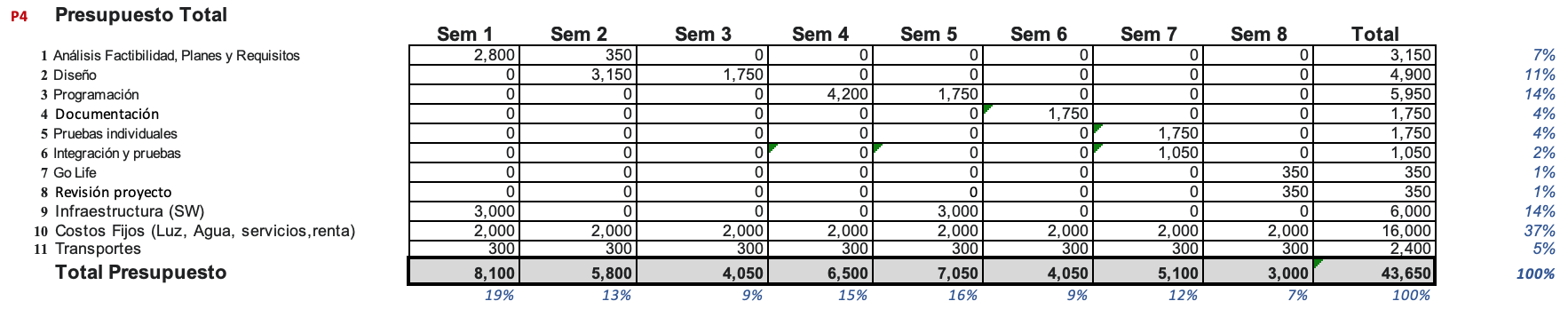
Definiendo a 350 la hora de programación tenemos como tabla 2:



Ahora, estimando los costos, tenemos:



El presupuesto total queda de la siguiente manera:



El total del proyecto saldría en 43,650 a terminar en un plazo máximo de 8 semanas.

# Desarrollo

Para llevar a cabo el proyecto, vamos a necesitar unos archivos llamados headers que se van a importar con la directiva *include*, por lo tanto, tenemos la siguiente tabla:

|  |  |
| --- | --- |
| **Header** | **Descripción** |
| glfw3.h | Es una biblioteca de utilidad ligera para uso con OpenGL. Proporciona a los programadores la capacidad de crear y dirigir ventanas y aplicaciones OpenGL, así como recibir la entrada de joystick, teclado y ratón |
| stdlib.h | Es el archivo de cabecera de la biblioteca estándar de propósito general del lenguaje de programación C. Contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras. |
| glm.hpp | Es una librería matemática escrita en C++ para el desarrollo de software gráfico basado en OpenGL. Particularmente, glm permite trabajar con vectores, matrices y funciones. |
| matrix\_transform.hpp | Define funciones que generan transformaciones comunes de matrices. |
| type\_ptr.hpp | Maneja la interacción entre punteros y vectores, tipos de matriz. Esta extensión define una función sobrecargada, glm :: value\_ptr. Vuelve un puntero al diseño de memoria del objeto. Los tipos de matriz almacenan sus valores en orden de columna principal. Esto es útil para cargar datos en matrices o copiar datos en objetos de búfer. |
| stb\_image.h | Es una cabecera que se utiliza para cargar imágenes. |
| SDL.h | SDL es una biblioteca multiplataforma (Linux, Windows, ...) para el control multimedia del ordenador. Entre otras cosas permite controlar los sistemas de vídeo y audio y nos da la posibilidad de gestionar los eventos del sistema (pulsaciones de teclas, movimientos de ratón o joystick, etc). |
| shader\_m.h | Archivo que mandará instrucciones al fragment shader. |
| camera.h | Cabecera que se encarga del manejo de la cámara dentro de nuestro entorno. |
| modelAnim.h | Cabecera que se encarga del manejo de los modelos que son animados. |
| model.h | Cabecera que se encarga del manejo de los modelos que no son animados. |
| Skybox.h | Cabecera que se encarga del manejo de los skybox |
| iostream | Es un componente de la biblioteca estándar (STL) del lenguaje de programación C++ que es utilizado para operaciones de entrada/salida |

Ya con esos headers, tendremos todo lo necesario para poder comenzar el desarrollo del programa.

### Modelos

Para la carga de los modelos en el SO que estoy utilizando, debo agregar la ruta absoluta. Dichos modelos fueron descargados y editados en el software de modelado de Blender.

// load models

// -----------

Model piso("/Users/luiscarranza/Documents/CompuGrafica/proyectoFinal/ProyectoFinal/extraFiles/resources/objects/piso/Piso.obj");

Model carro("/Users/luiscarranza/Documents/CompuGrafica/proyectoFinal/ProyectoFinal/extraFiles/resources/objects/Lambo/Carroceria.obj");

Model llanta("/Users/luiscarranza/Documents/CompuGrafica/proyectoFinal/ProyectoFinal/extraFiles/resources/objects/Lambo/wheel.obj");

Model autoVerde("/Users/luiscarranza/Documents/CompuGrafica/proyectoFinal/ProyectoFinal/extraFiles/resources/objects/LamboRojo/Carroceria.obj");

Para poder crear un modelo usaremos la palabra reservada Model seguido de un identificador, que generalmente es el nombre, seguido de la ruta del archivo .obj.

Después de cargar un modelo, necesitamos ubicarlo en el escenario, por lo que podemos usar las siguientes líneas de código:

staticShader.use();

staticShader.setMat4("projection", projection);

staticShader.setMat4("view", view);

model = glm::mat4(1.0f);

model = glm::translate(model, glm::vec3(0.0f, -1.75f, 0.0f));

model = glm::scale(model, glm::vec3(0.2f));

staticShader.setMat4("model", model);

piso.Draw(staticShader);

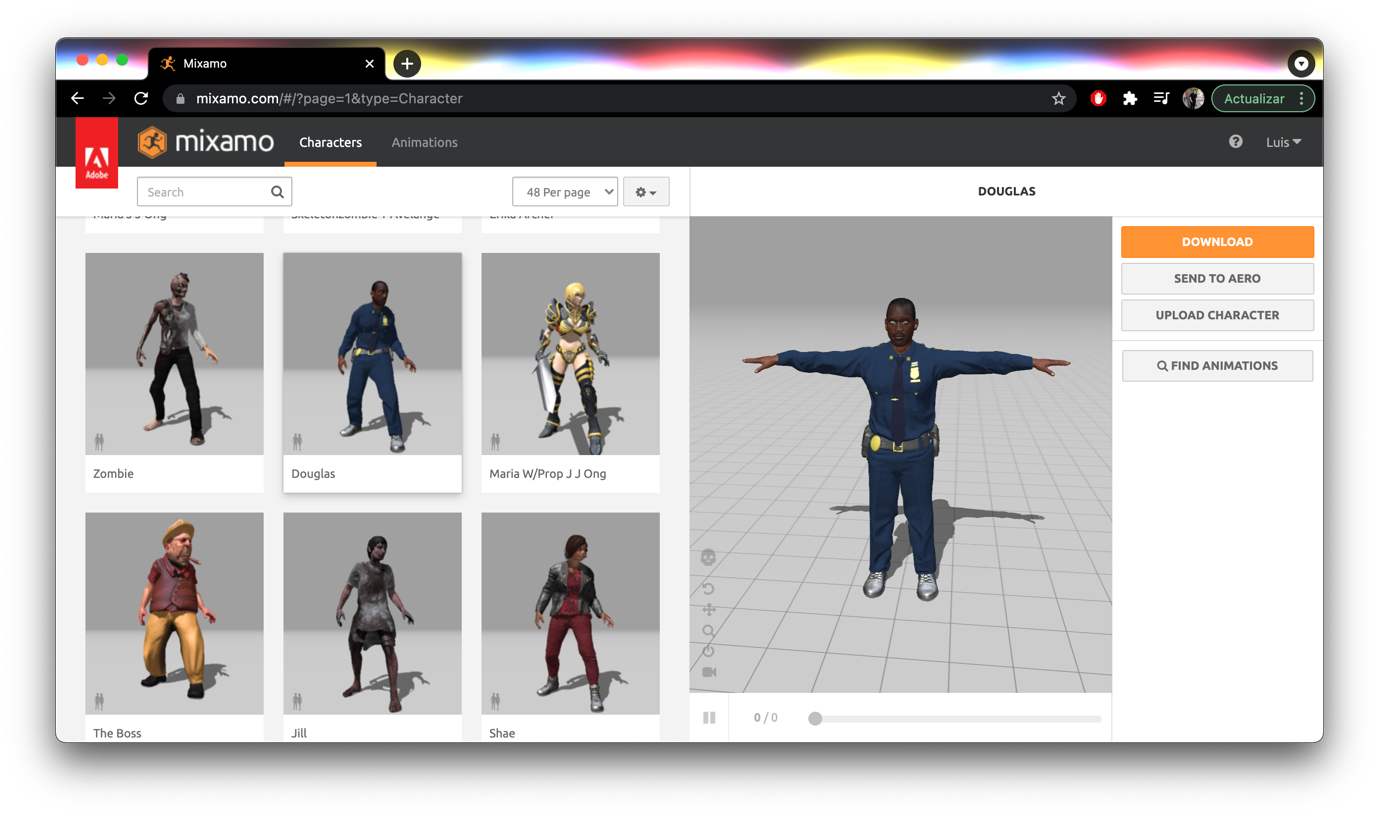
Inicializamos el staticShader para que pueda dibujar los objetos y seguido de eso, vamos a crear una matriz llamada model a la cual vamos a llenar con el valor de 1.0. Una vez teniendo esa matriz, lo que hacemos es modificar los valores de la matriz dependiendo de lo que nosotros querramos, por ejemplo, en el código anterior a la matriz model le aplicamos una translación en el eje *y* de -1.75, una escala de 0.2 que es el tamaño y por ultimo mandamos a dibujar la matriz con la línea staticShader.setMat4(“model”,model); piso.Draw(staticShader);

#### Tabla Modelos

|  |  |  |
| --- | --- | --- |
| Archivo | Descripción | Objeto |
| Alfombra | Alfombra que irá debajo del comedor. |  |
| sofa | Sofa que va en el dentro del departamento. |  |
| tele | Televisión de la sala. |  |
| comedor | Comedor que va entre la sala y la cocina. |  |
| cocina | La cocina del departamento.  Link de descarga:  https://www.turbosquid.com/3d-models/kitchen-baked-scene-3d-model-1611240 |  |
| refri | El refrigerador de la cocina. |  |
| mueble | Mueble que está a un lado del sofá. |  |
| muebleTV | Mueble donde se pondrá la televisión.  Link de descarga:  https://www.turbosquid.com/3d-models/3d-model-desk-console-1202445 |  |
| EdificioSeis | Edificio departamental donde está ubicado nuestro departamento.  Link de descarga:  https://www.turbosquid.com/3d-models/apartment-building-6-3d-model-1435007 |  |
| puertaEdificio6 | Estas son las puertas del edificio 6.  No hay link de descarga porque fue hecha por mi y mi equipo. | Nota, son dos puertas distintas. |
| puertaCorredizaEdificio6 | Esta es la puerta ancha del edificio seis. Esta se encuentra a un lado de las ventanas que se abren.  No hay link de descarga porque fue hecha por mi y mi equipo. |  |
| ventanaEdificio6 | Son las ventanas que se abren y cierran para salir al balcón del edificio seis.  No hay link de descarga porque fue hecha por mi y mi equipo. |  |
| mesa | Es la mesa que se encuentra en la azotea. |  |
| silla | Son las sillas que van acompañando la mesa de la azotea |  |
| aplauso | Es el nombre del modelo que tiene la animación que va aplaudiendo en la silla izquierda de la mesa de la azotea. |  |
| sentado | Es el nombre del modelo que tiene la animación que está sentado en la silla derecha de la mesa de la azotea. |  |
| carro | Es el auto de color amarillo que se ve en la calle. |  |
| llanta | Es la llanta de los autos. |  |
| autoVerde | Es el auto verde que se ve en la calle. |  |
| carroP | Es el modelo del auto gris sin puerta. Sobre ese modelo vamos a montar |  |
| puerta | La puerta del auto gris. |  |
| kiosco | Es el kiosco del parque. |  |
| kate | Modelo Anim humanoide que tiene una animación para caminar. |  |
| personajeHombre | Modelo Anim humanoide que tiene una animación para caminar. |  |
| megan | Modelo Anim humanoide que tiene una animación para caminar. |  |
| consulado | Edificio departamental de color blanco. |  |
| edificioCuatro | Edificio departamental que tiene tienda en la planta baja. |  |

### Modelos animados

Para el uso de este tipo de modelos usaremos la página web de mixamo, donde podemos encontrar modelos articulados con animaciones predefinidas, como se muestra a continuación:



Para poder ocupar modelos de este tipo, debemos descargarlos en formato \*.dae. Estos modelos de cargan de una manera un poco diferente.

ModelAnim megan("/Users/luiscarranza/Documents/CompuGrafica/proyectoFinal/ProyectoFinal/extraFiles/resources/objects/megan/megan.dae");

megan.initShaders(animShader.ID);

ModelAnim kate("/Users/luiscarranza/Documents/CompuGrafica/proyectoFinal/ProyectoFinal/extraFiles/resources/objects/kate/kate.dae");

kate.initShaders(animShader.ID);

La diferencia radica en que debemos mandar a llamar al identificador del objeto anim, e invocar el método initShaders(animShader.ID);

De igual manera podremos aplicarle operaciones al modelo y darle su ubicación en el escenario.

model = glm::translate(glm::mat4(1.0f), glm::vec3(-46.0f,0.0f,

-80.0f+movPersonaje));

model = glm::scale(model, glm::vec3(0.069f));

animShader.setMat4("model", model);

kate.Draw(animShader);

Nótese que se manda a dibujar el animShader en lugar del staticShader.

### Casos de uso animaciones

Puertas y ventas del departamento

|  |  |
| --- | --- |
| **Caso de Uso** | Puertas Edificios |
| **Actor** | Usuario final de la interfaz gráfica |
| **Propósito** | Abrir y cerrar las puertas y ventanas del departamento, con la tecla O (open) y C (close). |

Movimiento Autos, personas

|  |  |
| --- | --- |
| **Caso de Uso** | Autos, personas, puerta auto |
| **Actor** | Usuario final de la interfaz gráfica |
| **Propósito** | Con la tecla de espacio, las personas caminarán hacia delante, el auto amarillo avanzará hacia delante y dará una vuelta a la izquierda, se abrirá y cerrará la puerta del auto gris. |

Reinicio animaciones

|  |  |
| --- | --- |
| **Caso de Uso** | Reinicio animaciones |
| **Actor** | Usuario final de la interfaz gráfica |
| **Propósito** | Con la tecla R vamos a regresar todas las animaciones a como estaban antes de activarlas. |

### Diccionario de datos

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Tipo de dato** | **Descripción** |
| MovementSpeed | Float | Variable que controla la velocidad de la cámara |
| movAuto\_x | Float | Variable que nos ayudará a mover el auto amarillo en el eje x |
| movAuto\_z | Float | Variable que nos ayudará a mover el auto amarillo en el eje z |
| movAuto\_y | Float | Variable que nos ayudará a mover el auto amarillo en el eje y |
| Orienta | Float | Variable que nos ayudará a rotar el auto amarillo. |
| giroPuerta | Float | Variable que nos ayudará a girar la puerta del auto gris. |
| giroLlantas | Float | Variable que nos ayudará a realizar los giros de las llantas en el auto amarillo. |
| movPersonaje | Float | Variable que nos ayudará a realizar los movimientos en los personajes ModelAnim. |
| movPuerta | Float | Variable que nos ayudará a realizar los movimientos de las puertas y ventanas que se abren y cierran. |
| animacion | Bool | Variable que nos ayuda a iniciar la animación del movimiento del auto amarillo y la puerta del auto gris. |

|  |  |  |
| --- | --- | --- |
| animacionDos | Bool | Variable que nos ayuda a iniciar la animación de las puertas y ventanas que se abren y cierran. |
| puertaFlag | Bool | Variable que nos ayuda a detener la puerta en una cierta posición para después poder cerrar la puerta. |
| flag | Bool | Variable que nos ayuda al control de los condicionales en las animaciones. |
| piso | Model | Objeto de tipo Model que se encarga de simular el piso del escenario. |
| carro | Model | Objeto de tipo Model que se encarga de simular el auto amarillo del escenario. |
| llanta | Model | Objeto de tipo Model que se encarga de simular las llantas del auto amarillo, verde y gris del escenario. |
| autoVerde | Model | Objeto de tipo Model que se encarga de simular el auto verde del escenario. |
| carroP | Model | Objeto de tipo Model que se encarga de simular la puerta del auto gris del escenario. |
| puerta | Model | Objeto de tipo Model que se encarga de simular las puertas del departamento que estamos ambientando. |
| arbol | Model | Objeto de tipo Model que se encarga de simular los arboles escenario. |
| mesa | Model | Objeto de tipo Model que se encarga de simular la mesa que se encuentra en la azotea del edificio que estamos ambientando. |
| silla | Model | Objeto de tipo Model que se encarga de simular las sillas de la mesa de la azotea. |
| puertaEdificio6 | Model | Objeto de tipo Model que se encarga de simular la puerta del edificio seis. |
| puertaCorredizaEdificio6 | Model | Objeto de tipo Model que se encarga de simular la puerta corrediza del edificio seis. |
| ventanaEdificio6 | Model | Objeto de tipo Model que se encarga de simular la ventana del edificio seis. |
| edificioCuatro | Model | Objeto de tipo Model que se encarga de simular el edificio cuatro. |
| Consulado | Model | Objeto de tipo Model que se encarga de simular el edificio llamado consulado. |
| edificioSeis | Model | Objeto de tipo Model que se encarga de simular el edificio seis. |
| kiosco | Model | Objeto de tipo Model que se encarga de simular el kiosco. |
| personajeHombre | ModelAnim | Objeto de tipo ModelAnim que se encarga de simular a un hombre que camina por la banqueta. |
| megan | ModelAnim | Objeto de tipo ModelAnim que se encarga de simular a una mujer que camina por la banqueta. |
| kate | ModelAnim | Objeto de tipo ModelAnim que se encarga de simular a una mujer que camina por la banqueta. |
| sentado | ModelAnim | Objeto de tipo ModelAnim que se encarga de simular a una mujer sentada en la mesa del edificio seis. |
| aplauso | ModelAnim | Objeto de tipo ModelAnim que se encarga de simular a una mujer sentada dando aplausos en la mesa del edificio seis. |
| sofa | Model | Objeto de tipo Model que se encarga de simular el sofá del departamento. |
| tele | Model | Objeto de tipo Model que se encarga de simular la televisión del departamento. |
| comedor | Model | Objeto de tipo Model que se encarga de simular el comedor del departamento. |
| cocina | Model | Objeto de tipo Model que se encarga de simular la cocina del departamento. |
| refri | Model | Objeto de tipo Model que se encarga de simular el refri del departamento. |
| mueble | Model | Objeto de tipo Model que se encarga de simular el mueble del departamento. |
| muebleTv | Model | Objeto de tipo Model que se encarga de simular el mueble donde estará la TV del departamento. |

### Diccionario de funciones

|  |  |  |  |
| --- | --- | --- | --- |
| **Identificador** | **Valor de retorno** | **Parámetros** | **Descripción** |
| animate | void | void | Esta función se encarga de llevar el control de animaciones. |
| main | int | - | Esta función es la función principal y es donde mandaremos a llamar a todas las demás funciones. Es la función que se ejecutará primero |
| my\_input | void | Apuntador de tipo GLFWwindow, int, int, int, int. | En esta función vamos a llevar el manejo de las teclas del teclado. Realizaremos diferentes funciones dependiendo de la tecla presionada. |
| mouse\_callback | void | Apuntador de tipo GLFWwindow, double, double | Función que se encarga del manejo del ratón. |
| scroll\_callback | void | Apuntador de tipo GLFWwindow, double, double | Esta función se activa cuando hacemos scroll con el ratón. |

# Proyecto en Github

El proyecto fue alijado en la plataforma de Github para poder tener un mejor control de las versiones y no perder información importante. En el siguiente link puedes encontrar todo el proyecto completo.

<https://github.com/lucalice/PFT-CGeIHC_LuisEnriqueCarranzaEscobar.git>

# Referencias Modelos

cocina: <https://www.turbosquid.com/3d-models/kitchen-baked-scene-3d-model-1611240>

edificioSeis: <https://www.turbosquid.com/3d-models/apartment-building-6-3d-model-1435007>

edificioCuatro: <https://www.turbosquid.com/3d-models/apartment-house-building-3d-model-1650425>

mueble: <https://www.turbosquid.com/3d-models/max-tv-unit/536459>

muebleTV: <https://www.turbosquid.com/3d-models/3d-model-desk-console-1202445>

refri: <https://www.turbosquid.com/3d-models/fridge-3d-model-1722017>