Asignatura: Informática Gráfica (Prácticas)

Curso: 2022-2023

Etapas

Etapa 1: Generación del Workspace y primer programa en Opengl mediante GLUT

Compilar la Etapa1 sobre Linux

Abrir la carpeta home: (icono Inicio situado en el escritorio), en esta carpeta ubicaremos nuestros ficheros. Crear un fichero con nombre etapa1.c: clicando con el botón derecho en el área vacía de la carpeta (fondo blanco) seleccionar la opción "Crear fichero de texto", nombrarlo como etapa1.c Copiar el código fuente de la etapa1 en el fichero etapa1.cpp: Abrir el fichero de código fuente mediante un clic del ratón, copiar de la página web, pegar en el fichero etapa1.c y grabar.

Abrir una consola de comandos: para la instalación actual, clicar en la K (parte inferior izquierda de la pantalla) seleccionar Debian->Shells para X->Konsole. Compilar la práctica: ejecutar el siguiente comando en la consola "cc -o etapa1 etapa1.c -lglut". El comando invoca al compilador de C para que compile el fichero etapa1.c genere como salida el fichero etapa1 y añada al linkar la librería GLUT.

Para ejecutar realizar un clic sobre el fichero etapa1 ubicado en el directorio home.

Compilar la Etapa1 sobre Windows

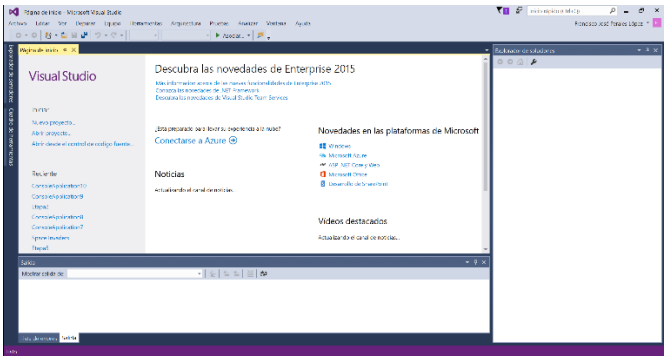
Como crear una Proyecto

Las practicas se desarrollaran con la herramienta Microsoft Visual Studio 20XX o similar, en esta herramienta trabaja con soluciones, que es un entorno de trabajo. Nuestro primer paso será crear una solución con un proyecto, a esta solución iremos añadiendo nuevos proyectos que corresponderán con cada una de las etapas de la práctica.

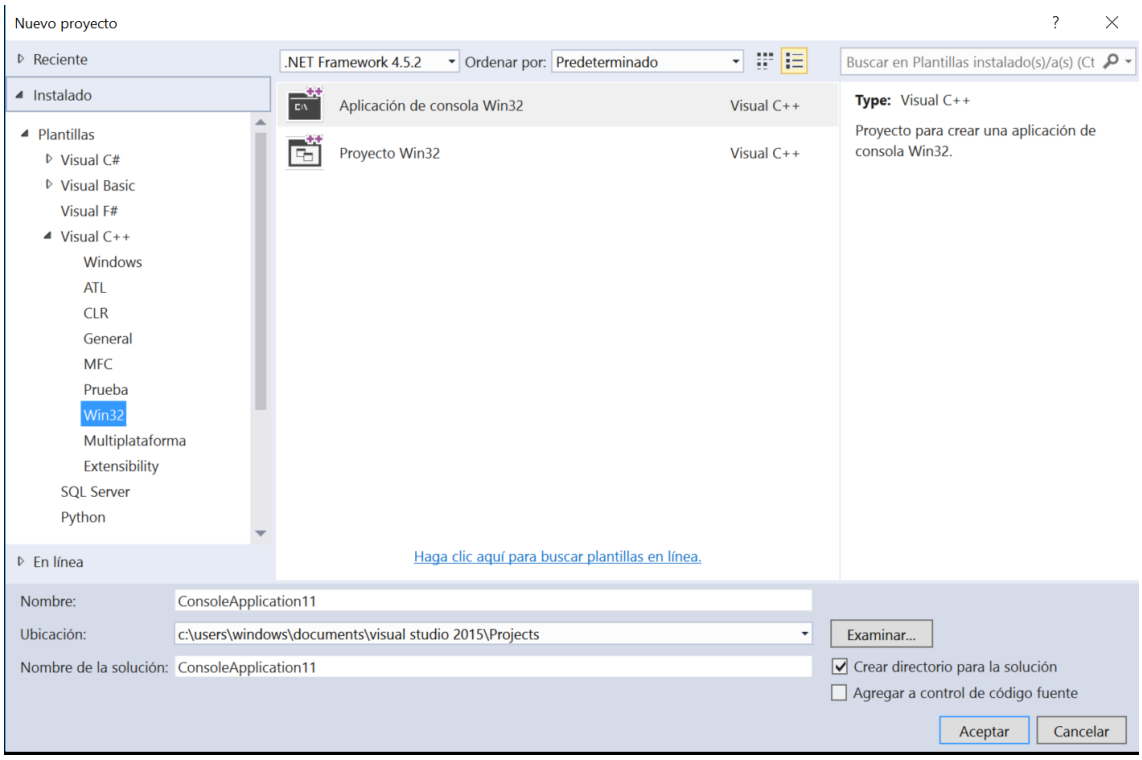
Nota. Se deja libertad al estudiante de buscar cualquier otro compilador y/u otro lenguaje de programación siempre y cuando se cumplan los objetivos definidos en las prácticas para la implementación de los conceptos teóricos vistos en clase.

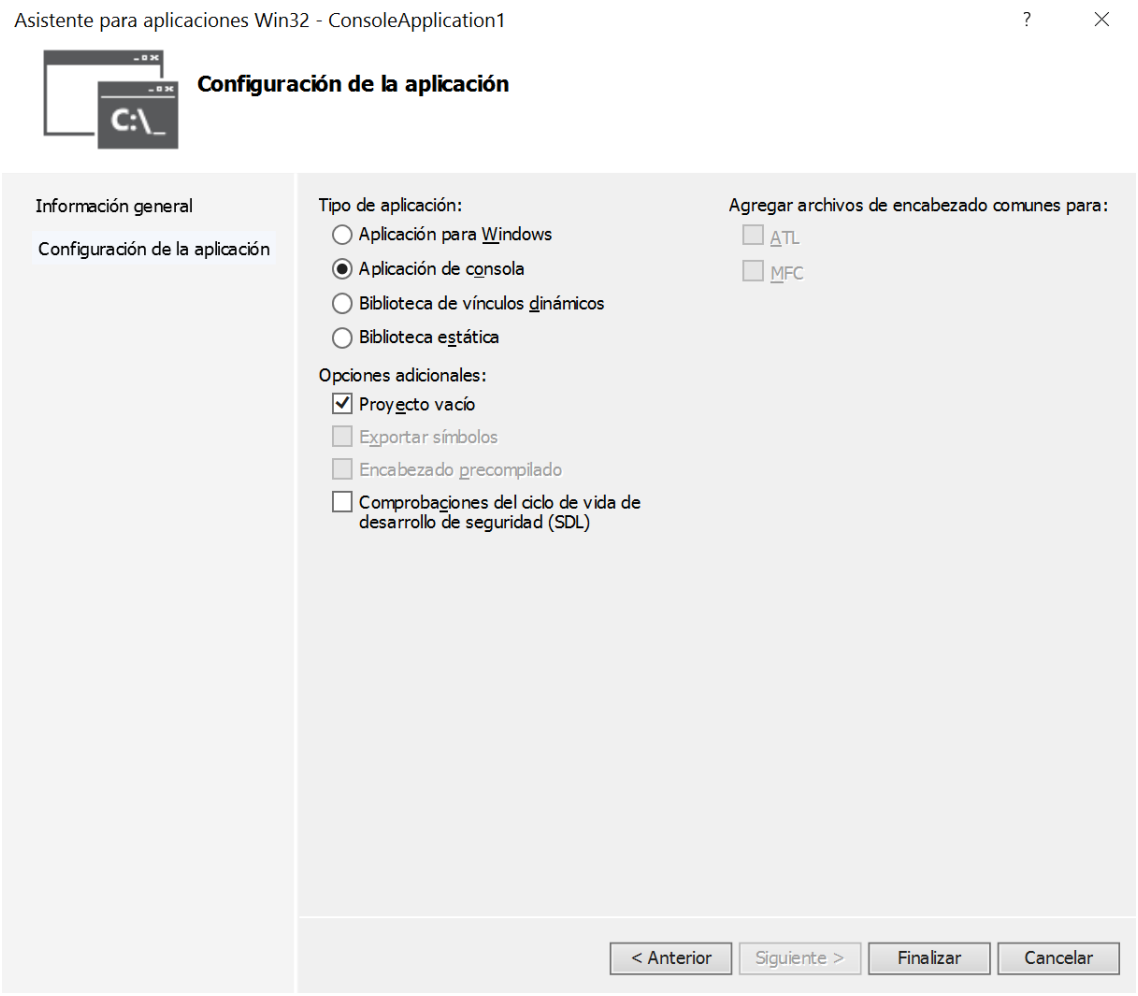
Como añadir un proyecto a una Solución

A continuación veremos cómo crear el proyecto correspondiente a la etapa1. Primero iremos a Archivo/Nuevo/Proyecto. Seleccionaremos dentro de Visual C++ un proyecto de tipo Aplicación de Consola Win32, como nombre del proyecto ponemos Practica1 y como nombre de la solución PracticasIG. Con esto creamos una solución que incluye un proyecto, a medida que vayamos avanzando crearemos un proyecto para cada etapa.



Nos aparece un asistente, seleccionamos Aplicación de Consola y Proyecto Vacío.

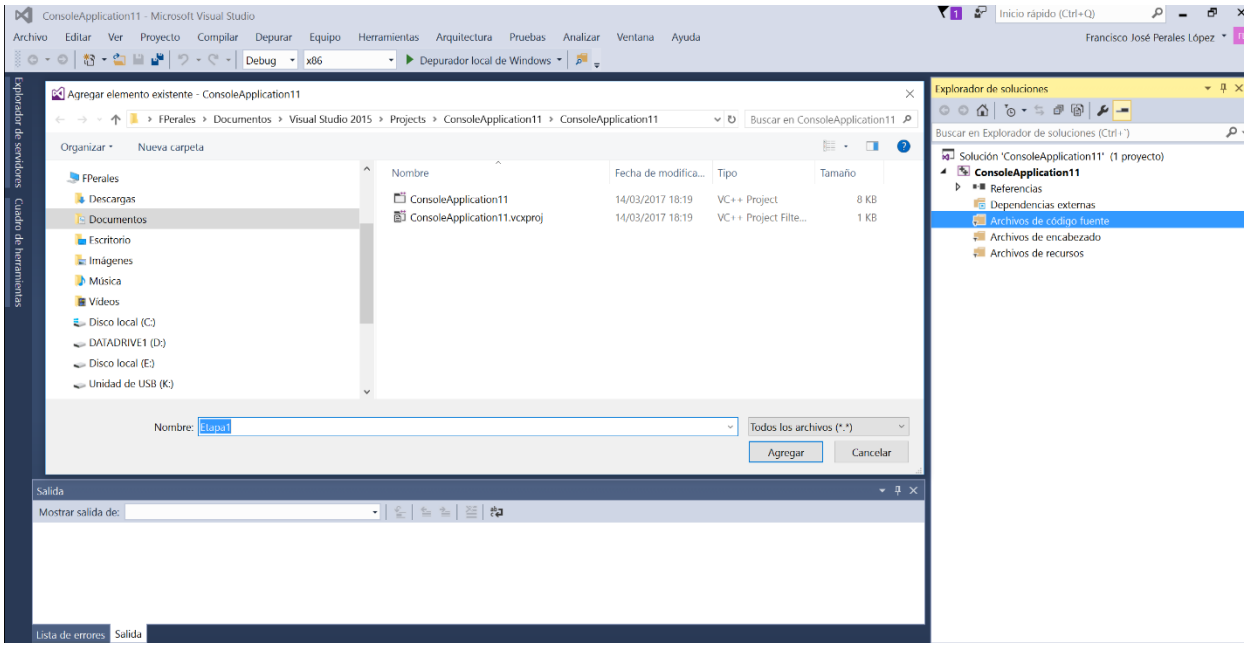




Como añadir fichero fuente a un proyecto

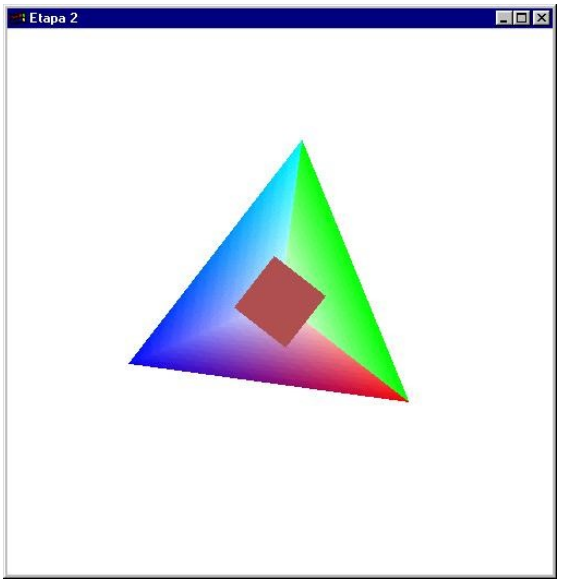
Ya tenemos creada la solución y nuestro proyecto, ahora debemos incluir un fichero C para que podamos compilar. Clicamos con el botón derecho en el proyecto, seleccionamos Agregar/Nuevo Elemento o bien si ya lo tenemos añadir elemento existente. Nos aparecerá otro asistente, seleccionamos Archivo C (.cpp), en el nombre ponemos Practica1.cpp

En este nuevo fichero (etapa1.c) escribiremos nuestro programa que usará la librería GLUT, es necesario tener copiada la librería glut32.lib en el directorio dónde se encuentra el resto de librerías, así como el ficheros cabecera glut.h en el directorio include. El fichero glut32.dll debe estar colocado en alguna carpeta del path, para que cuando se ejecute nuestro programa la encuentre (p.e. C:\windows). Copiar el código fuente de la etapa 1 (otro fichero en Moodle).



Una vez tenemos el código podemos ejecutarlo de dos formas diferentes:

- Modo Debug (F5), en este modo recibiremos en la ventana de Output los mensajes de error y los que generemos nosotros.
- Modo Execute (Ctrl+F5), en este modo no recibimos información, aunque el código de ejecución es el mismo, nuestra aplicación y Visual C no están conectados.
- Resultado de la Etapa 1 es la imagen siguiente.



Para usuario de Mac existe la versión JOLG en NetBeans. Se puede usar también ese entorno, siempre que se cumplan los objetivos de las prácticas. Para descargar JOGL ir a http://jogamp.org/deployment/jogamp-current/archive/

Etapa 2: Doble buffer y escalado

Doble Buffer

En el programa anterior de la animación, las primitivas se realizan directamente sobre el buffer del dispositivo de pantalla por lo que se produce parpadeo al borrar la ventana ya continuación redibujarla, para evitar este efecto se utiliza un doble buffer, se visualiza sobre un segundo buffer y una vez se obtiene la imagen deseada se intercambia con el buffer de pantalla.

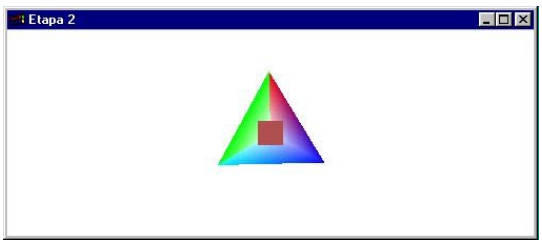
Las funciones a utilizar son:

```
glutInitDisplayMode (GLUT_RGBA | GLUT_DOUBLE);
```

```
glutSwapBuffers();
```

Mantenimiento de aspect/ratio.

Al cambiar el tamaño de la ventana, el cuadrado se deforma porque el volumen de recorte definido permanece intacto, corregir el código de tal manera que cuando la ventana cambie de tamaño se ajuste a las proporciones de un cuadrado (alto=ancho), para verificar mejor podemos insertar un cuadrado.



Capturamos el evento de cambio de tamaño de ventana, al darse este evento modificamos el volumen de visualización para que tenga las mismas proporciones que la ventana. La declaración de la función GLUT donde registramos el evento de cambio de ventana es:

```
void glutReshapeFunc(void (*func)(int width, int height));
```

Una vez realizado el ajuste de escalado para evitar la distorsión, hemos de implementar una escena 2D, con los ejes de coordenadas centrados en el punto medio de la ventana. Sobre esta escena visualizaremos una serie de primitivas 2D en cada cuadrante y les aplicaremos las transformaciones básicas vistas en el tema 2. Definiremos al menos una transformación compuesta que implique combinaciones de las básicas (rotación sobre un punto fijo, escalado sobre un punto fijo, etc.)

Finalmente diseñaremos una estructura articulada plana (2D) simple, del estilo de un péndulo doble o péndulo caótico para poder estudiar el manejo de las transformaciones anidadas y el uso de las funciones push y pop de la pila de matrices para las transformaciones.

La estructura simple del péndulo serán polígonos rectangulares. Se aplicaran a todos los elementos de la escena aspectos de color y atributos, de cara a practicar con todas las opciones de las primitivas básicas. Estructuras articuladas planas similares o equivalentes son aceptadas (grúas, gusano/snake, balancines, brazos robóticos, etc.)

Se puede ya definir de forma preliminar la escena final de la etapa 6, incluyendo la lámpara articulada 2D y unas primitivas redondas (círculo, elipse, etc.)

Recomendación: Visualizar ejes de referencias (ON/OFF) y trayectorias de objetos.

Nota. No pasar a etapa 3 sin dominar perfectamente todas las transformaciones 2D y los aspectos de atributos de OpenGL primitivas y funciones básicas.

(Punto de Valoración Parcial Orientativo)

Etapa 3: Escena 3D

En esta etapa pasamos al espacio 3D. El objetivo es similar a los pasos de la etapa 2 pero ahora definimos un sistema de coordenadas cartesiano con los ejes (X, Y, Z). Tendremos que definir las funciones de ejes, y de planos de referencias que pueden ser visibles o no a petición del usuario (o con cierta transparencia, alpha cannel). El objetivo de esta etapa es crear una escena simple con primitivas 3D y aplicar las funciones de transformación y visualización del tema 2 y 3.

Para ello:

- Añadir una 3ª coordenada z.
- Incluir primitivas de definición de objeto glutSolidSphere glutWireSphere de la librería GLUT o bien definir de forma explícita las propias primitivas 3D, o importar objetos de otros formatos.

```
glutSolidCube glutWireCube
glutSolidCone glutWireCone
glutSolidTorus glutWireTorus
glutSolidDodecahedron glutWireDidecahedron
glutSolidOctahedron glutWireOctahedron
glutSolidTetrahedron glutWireTetrahedron
glutSolidIcosahedron glutWireIcosahedron
glutSolidTeapot glutWireTeapot
```

- Definir una perspectiva con glFrustrum, comparar con glOrtho

```
glMatrixMode (GL_PROJECTION)
glFrustum (...)
glPushMatrix()
glPopMatrix ()
```

- Activar el buffer de profundidad para darnos cuenta de la proximidad de los objetos

```
glEnable (GL_DEPTH_TEST)
glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
```

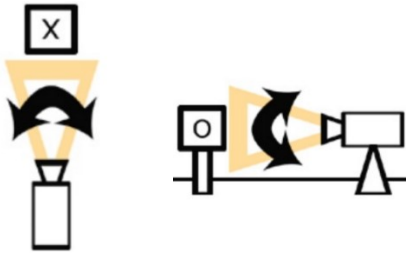
Comenzar a dibujar una escena simple con diversidad de primitivas de la glut.h y propias. Se puede ya definir de forma preliminar la escena final de la etapa 6, incluyendo la lámpara articulada 3D y unas primitivas redondas 3D (esfera, elipsoide, cubo, cilindro, etc.)

- **Opcional:** Crear tus propias primitivas 3D, o importar objetos de otras librerías.

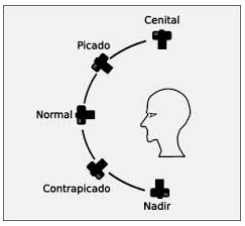
Etapa 4: Movimiento de Cámara

En esta etapa el objetivo es que el usuario pueda moverse libremente por la escena. Es decir la cámara virtual que nos da la vista de nuestra escena podrá variar de acuerdo con una seria de funciones. El movimiento de la cámara vendrá dado por las funciones siguientes de orientación y translación:

- Movimiento de la cámara de Paneo (la cámara gira sobre su propio eje de izquierda a derecha). LEFT Girar Izquierda/ RIGHT Girar Derecha
- Movimiento de la cámara de Tilt (la cámara gira sobre su propio eje de hacia arriba o hacia abajo)



- Movimientos laterales o acercamiento/alejamiento (travelling/Dolly(in/out))
- (Flecha UP Avanzar/DOWN Retroceder, IZQ/DER)
- Movimientos angulares de la cámara. Coordenadas Esféricas. Objetos en el centro de la escena. Rotación respecto al origen de la cámara. Ángulos: Nadir, Contrapicado, Normal, Picado y Cenital.




```
glutSpecialFunc (...)
```

Reemplazar la función glFrustrum por gluPerspective

```
gluPerspective (...)
gluLookAt (...)
```

Variantes: Permitir cambiar entre la cámara móvil y un número de cámaras fijas situadas en puntos de interés de la escena. Por ejemplo, definir varias vistas ortogonales, planta, alzado, perfil y perspectiva. Definir trayectorias de la cámara, etc…

Definir otros planos distintos a los considerados (plano subjetivo, inclinación lateral, etc.)

Aplicar la vista de cámara a la escena preliminar de la etapa 6 si se ha diseñado así.

Recomendación: Visualizar ejes/planos de referencias (ON/OFF) y trayectorias de objetos. Visualizar trazado de la cámara (ON/OFF).

Etapa 5: Luces y Materiales, Sombreado

En esta etapa iniciaremos el proceso de realismo de la escena, para ello añadiremos luces a la escena y materiales a los objetos. Realizaremos estos aspectos mediante los siguientes pasos:

- Habilitaremos la renderización con luz:

```
glEnable (GL_LIGHTING)
glEnable (GL_COLOR_MATERIAL)
glColorMaterial (GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE)
```

- Insertaremos una luz que moveremos por la escena con las teclas: A, Z, S, X, D, C

```
glEnable (GL_LIGHT0)
glLight (...)
glutKeyboardFunc (..)
```

- Probar con diferentes parámetros y características de los objetos y la luz: ambiente, difuso, especular glMaterial (...)

```
glLight (...)
```

- Añadiremos varias fuentes de luz y veremos sus efectos y permitiremos mediante teclado encender y apagar luces.

Finalmente permitiremos por teclado el cambio de modelo de sombreado.

OpenGL dispone de dos tipos de sombreado GL_FLAT y GL_SMOOTH.

- Elegir el tipo de sombreado y cambiarlo con la tecla espacio

```
glShadeModel (...)
```

- Cambiar la normal a alguna superficie definida y ver el efecto que produce

```
glNormal (...)
```

- Renderizar objetos en la escena que realicen algún movimiento. Aplicar la vista de cámara a la escena preliminar de la etapa 6 si se ha diseñado así.
- **Nota. No pasar a etapa 6 sin dominar perfectamente todas funciones básicas de visualización, iluminación y materiales.**

(Punto de Valoración Parcial Orientativo)

Etapa 6: Realismo I

Esta etapa debe recopilar todos los aspectos de las anteriores mediante la creación de una escena **"concreta"** lo más realista posible usando Opengl. Se debe incluir al menos **dos** de las opciones optativas:

- **Obligatorio:**
- Escena compleja con un mínimo de objetos 3D de los cuales algunos sean articulados y pueden ser de la GLUT y definidos propios.
- Opciones de menú/ratón para la manipulación de posición orientación de cámara, luces (ON/OFF) y varias vistas de la escena.
- **Optativo:**
- Texturas, Bump Mapping, Sombras, Anti-aliasing
- Niebla, Curvas y Superficies (NURBS)
- Otras librerías (SOIL, freeglut, OpenAL, etc...)

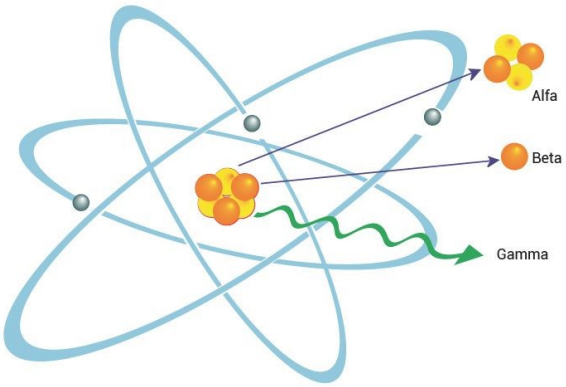
El contenido de la escena es común para todos y tiene que ajustarse a las imágenes incluidas del juego de cartas **GeniusUp Personajes e Inventos**. El contenido libre adicional añadirá elementos similares que aporten un aspecto avanzado dentro de los puntos optativos de la escena. Se puede realizar un video final de la escena creada donde se observe de forma clara todos los aspectos desarrollados en la práctica de la etapa final. La etapa final se podrá implementar de **3 formas diferentes:**

Opción 1: Continuar con las funciones básicas de Opengl sin shaders.

Opción 2: Implementar la etapa 6 con versiones de Opengl con shaders.

Opción 3: Usar el entorno Blender-Unity para incluir funcionalidades anteriores pedidas.

Se debe seleccionar un personaje de la baraja de Química/Informática/Matemáticas y modelar su avatar y definir una interacción del personaje con algún invento propio.



Objetivo: La escena 3D generada simularía una reacción de fisión o fusión o el efecto de la radiactividad sobre objetos cercanos.

Información adicional para la opción 3:

- Breve explicación de Blender y sus funciones
  https://www.youtube.com/watch?v=eFowqayoSKc
  https://www.youtube.com/watch?v=MusMqDWXaIc
  https://www.youtube.com/watch?v=PHhFFIzE53o
- Como insertamos un personaje hecho en blender a Unity
- Como animamos este personaje en Unity
  https://www.youtube.com/watch?v=pMH1igyAUVA
- Mostrar un ejemplo de cómo podemos manejar un objeto (sea de blender o de otro sitio) dentro de un escenario 3D (saltar, moverse..) Dos Horas de clase.

(Punto de Valoración Final Definitiva + Informe detallado)

Nota. Cada etapa se guarda en proyecto separado y se presenta un informe final de todas las etapas explicando lo implementado, ventajas e inconvenientes, manual de usuario y una **valoración conceptual critica** de lo aprendido al final del trabajo realizado.

Profesor: Dr. Francisco J. Perales

Palma 15/3/2023