Mayo 2019

OpenGL y Bullet:

Para el desarrollo de este proyecto, he utilizado el previo proyecto de 3D Avanzado, el cual constaba de una escena 3D usando OpenGL, a partir del mismo, he implementado la librería Bullet para poder mejorar el proyecto con físicas y hacerlo así más extenso y completo. A pesar de esto la implementación de Bullet ha requerido de ciertos nuevos métodos sobre el renderizado y los modelos del anterior proyecto, tales como añadir en cada modelo una referencia a un body y una shape de Bullet, añadir una variable que permita que el modelo no se visualice ni actualice o añadir un constructor y destructor de físicas cada vez que se cree un nuevo modelo 3D entre otros.

Modelos:

Los modelos, que desde el anterior proyecto poseían capacidades de ser dibujados mediante OpenGL a partir de un modelo3D, así como poseer una textura indicada por el usuario y ser situados y animados en el espacio, de forma directa o jerárquica, ahora poseen componentes que permiten su comportamiento de física simulada en el mundo, tales como un btRigidBody y un btCollisionShape, mediante el constructor de físicas del modelo, llamado setup\_body\_physics() podemos ajustar la forma de la collision shape así como las propiedades físicas del modelo.

Para que los modelos puedan reaccionar a la física, su matriz de modelo en el mundo se actualiza cada update con la matriz que Bullet actualiza con el comportamiento físico del modelo.

Escena:

El modelo de la escena ha tratado de ser lo más fiel posible a las especificaciones del proyecto, está compuesto por múltiples modelos 3D, las plataformas de la escena poseen limites en el desplazamiento y rotación en todos los ejes, lo que hace que estén congelados en el espacio, la plataforma móvil tiene desbloqueada el desplazamiento en el eje X y posee una animación que permite desplazarlo de un lado a otro. La llave posee una animación de rotación, si el jugador colisiona el vehículo con la llave, tanto llave como puerta se destruyen, lo que permite al jugador continuar. El objetivo a derribar no posee ninguna limitación espacial y puede ser derribado si lo impacta una esfera.

Catapulta:

El sistema de catapulta está formado por la unión de dos cuerpos rígidos con un btHinge, dicho btHinge posee un clamp en ángulo, lo que permite limitar el movimiento del motor presente en el btHinge a lo necesario para poder propulsar la esfera simulando el comportamiento de una catapulta, dicho motor se activa pulsando la tecla <Espacio>, lo cual aplica una fuerza en dicho motor, si no se pulsa dicha tecla, el motor posee una pequeña fuerza aplicada que permite que vuelva a su posición original.

Esta catapulta permite lanzar esferas, dichas esferas se sitúan en la posición de la pala de la catapulta cada vez que se pulse <Espacio> (Siempre dentro de un tiempo limitado de 1s entre lanzamiento, aun así, si la esfera cae fuera del mapa, también puede volver a dispararse instantáneamente), la esfera se reposiciona siempre en dicha posición (No se instancian nuevas esferas), si dicha esfera cayese de la escena una vez lanzada, dejaría de dibujarse y actualizarse al ponerse a false el booleano is\_visible.)

Uso de motores:

Otro uso de motores, además del ya explicado uso para la catapulta, es en cada una de las ruedas del vehículo, las cuales poseen motores independientes, con capacidad de ajuste de dirección (Steering) con las teclas A/D y ajuste de potencia (Drive) con W/S, dichos motores forman parte de un bt2Hinge entre el chasis y cada rueda.

Apertura de puerta con llave:

La colisión del vehículo del jugador con la llave permite abrir la puerta azul translúcida que permite avanzar, dicha colisión además destruye tanto la puerta como la llave, la detección de dicha colisión se comprueba cada update, viendo si los cuerpos físicos con las IDs especificadas están colisionando en ese específico momento, la función que permite ver si ambos objetos están colisionando, llamada check\_collision(), utiliza la función de Bullet getNumManifolds(), para después comprobar cada colisión y se ver el UserIndex de cada objeto colisionando, si dichos objetos poseen los UserIndex especificados, se devuelve true al haberse encontrado una colisión entre esos elementos, si no, se devuelve false.