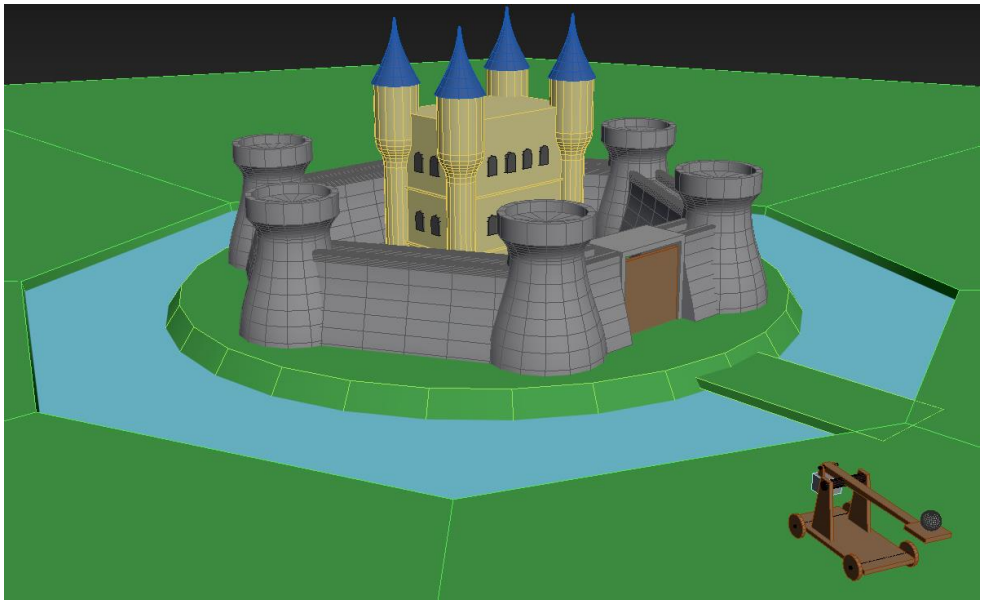
Sistemas Gráficos - TP1 – 2do cuat. 2022

Objetivo

Implementar la escena planteada en la siguiente figura, generando algunos de los modelos en forma paramétrica utilizando el algoritmo de superficies de barrido y de revolución junto con los algoritmos de curvas.

Algunos elementos como el portón y la catapulta tendrán animación controlada por el usuario.



La aplicación deberá contar con un menú dividido en al menos 3 bloques con las siguientes opciones:

- Castillo
  - cantidad de pisos (cambo)
  - ancho (slider)
  - largo (slider)
  - cantidad de lados de la muralla (slider entre 4 y 8)
  - altura de la muralla (slider)
  - actualizar (botón que regenera la escena)
- Escena
  - Apertura portón (slider q controla el ángulo entre 0 y 90)
  - Rotación Y de la catapulta (slider entre 0 y 360)
  - Disparar catapulta (botón que lanza la munición)
- Cámaras
  - Orbital general
  - Orbital catapulta
  - Primera persona
- Rendering
  - Default
  - Normales

El menú puede ser implementado con la librería dat.gui.

Cámaras

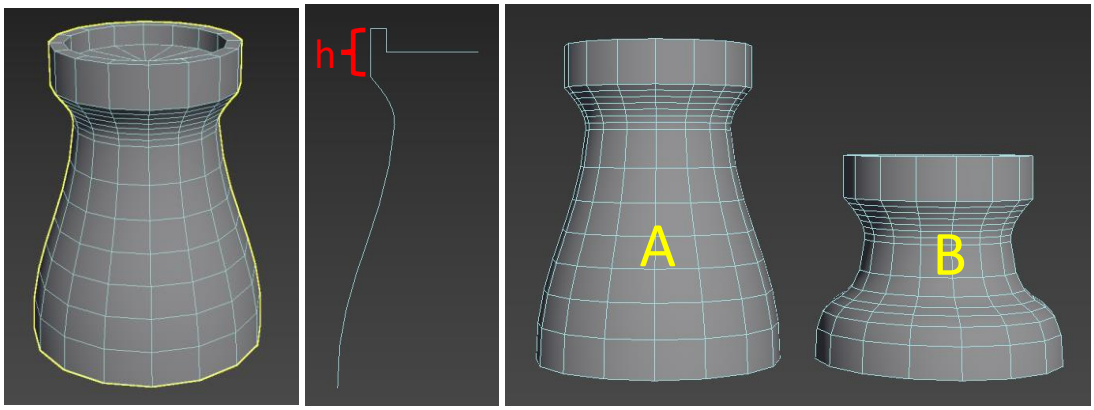
Se deberán implementar las siguientes cámaras

- Orbital general: centrada en el castillo
- Orbital catapulta: centrada en la catapulta
- Primera persona: simula una persona caminando sobre el plano del piso (usar las teclas ASDW para caminar y el mouse para rotar el punto de vista)

Las teclas 1,2,3 deberán permitir seleccionar las cámaras respectivas

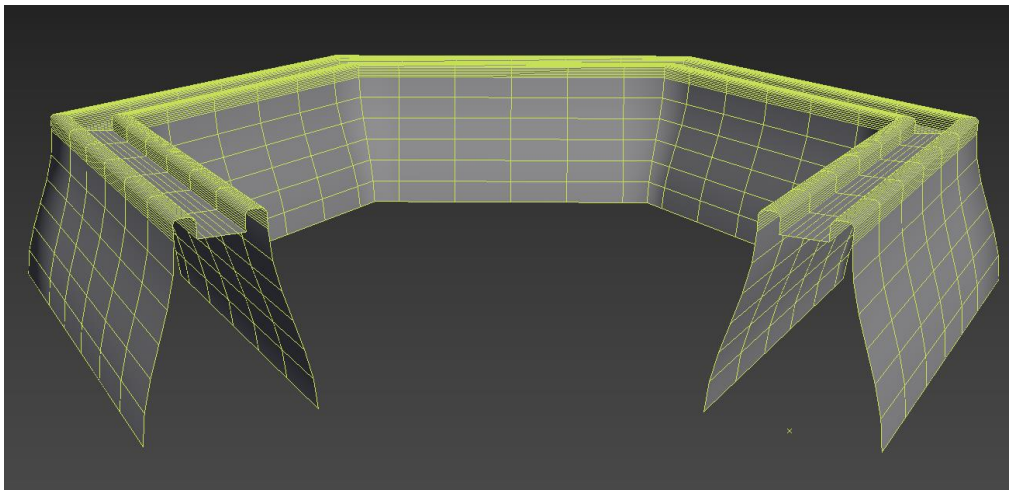
Torres de muralla

Las torres se deberán generar a partir de una curva Bezier cúbica y el algoritmo de superficies de revolución. Los puntos de control deberán reajustarse en función de la altura de la muralla. Esto significa que no se debe escalar simplemente el grafo de control verticalmente. Los puntos de control que corresponden a la parte superior deben desplazarse para mantener la altura "h", como se observa en los ejemplos A y B



Murallas

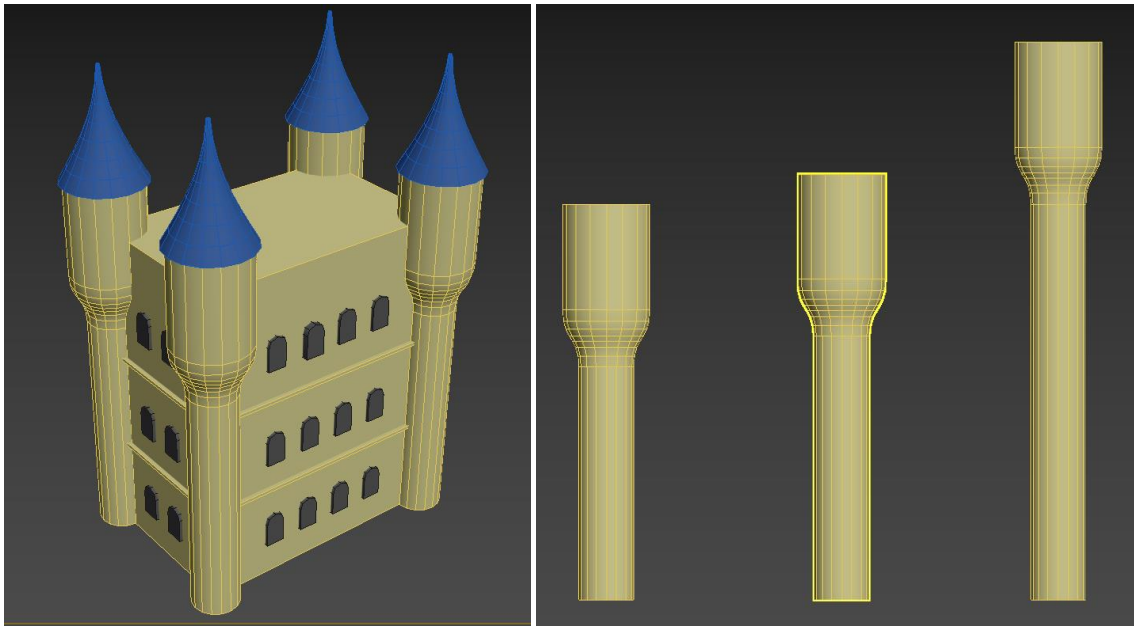
Igual que en el caso de las torres se deben ajustar los puntos de control en función de la altura de la muralla, manteniendo la altura "h" del balcón constante.



La malla puede implementarse como una única superficie de barrido como varios segmentos lineales individuales (4 a 8)

Castillo

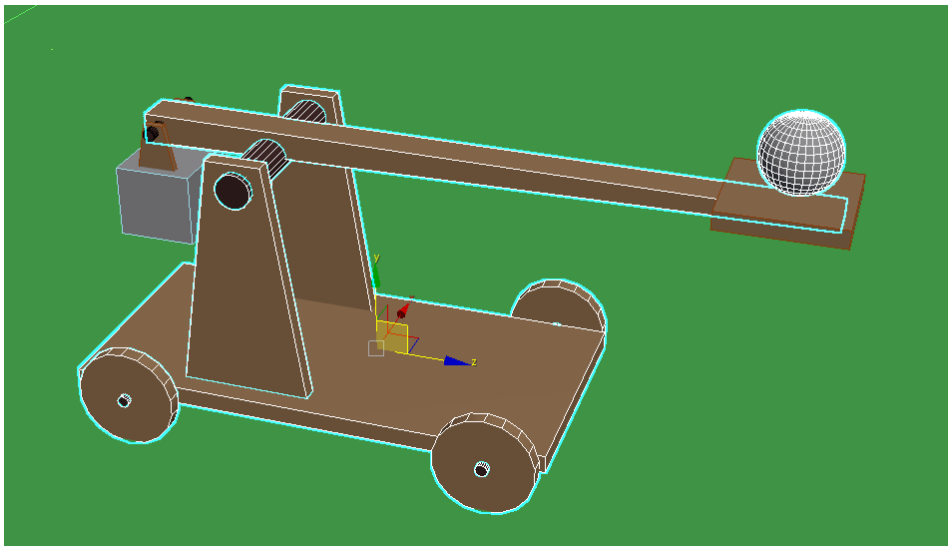
La cantidad de ventanas se deberá ajustar en función del ancho y largo. Se deberá establecer una constante “separación entre ventanas” y luego calcular cuantas caben en cada lado



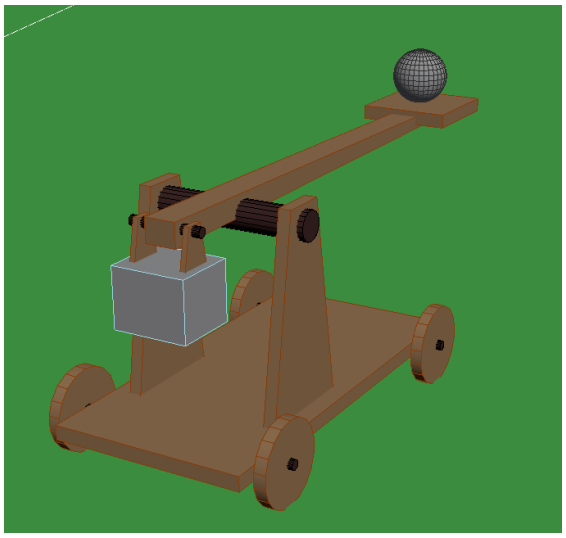
Las 4 torres deberán modelarse a partir de superficies de revolución y curvas de Bezier. igual que con las torres de las murallas, la sección mas gruesa de la torre tendrá una altura fija independiente de la altura de la torre. Se deberán reajustar los puntos de control en función de la cantidad de pisos del castillo. Las puntas de las torres deberán respetar la curvatura que se observa en la figura

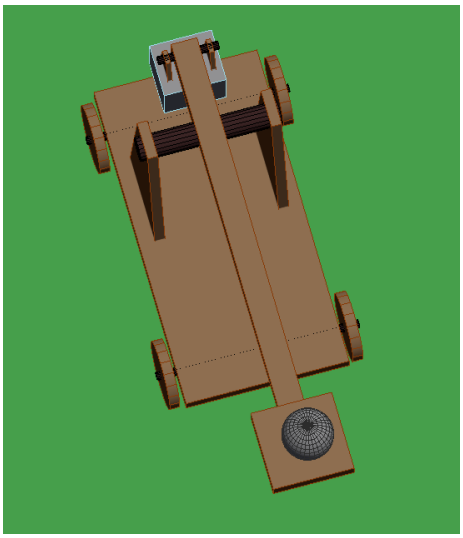
Catapulta

Deberá crearse una estructura jerárquica que represente la catapulta. La misma podrá dispararse desde el menú o con una tecla (opcional) y el usuario podrá controlar su orientación respecto del eje Y para apuntarla en distintas direcciones.



La munición deberá permanecer sobre la pala hasta el final del recorrido. Luego se separará siguiendo una trayectoria parabólica.
Eso se puede resolver planteando que la munición es un objeto “hijo” de la pala. En el momento en que se produce la separación se oculta ese objeto y se hace aparecer otro idéntico en su posición, pero no depende de ningún “padre”. La velocidad inicial y la posición definirá la trayectoria posterior





Desarrollo del trabajo

El TP se debe realizar en forma individual. Cada alumno debe programar su propio código. Deberán crear un repositorio privado en GitHub desde el comienzo e ir actualizándolo regularmente para poder evaluar el progreso del TP.

Deberán agregarme como colaborador para poder acceder al mismo. Además del link al repositorio se deberá proveer una URL para probar la aplicación funcionando desde el navegador.

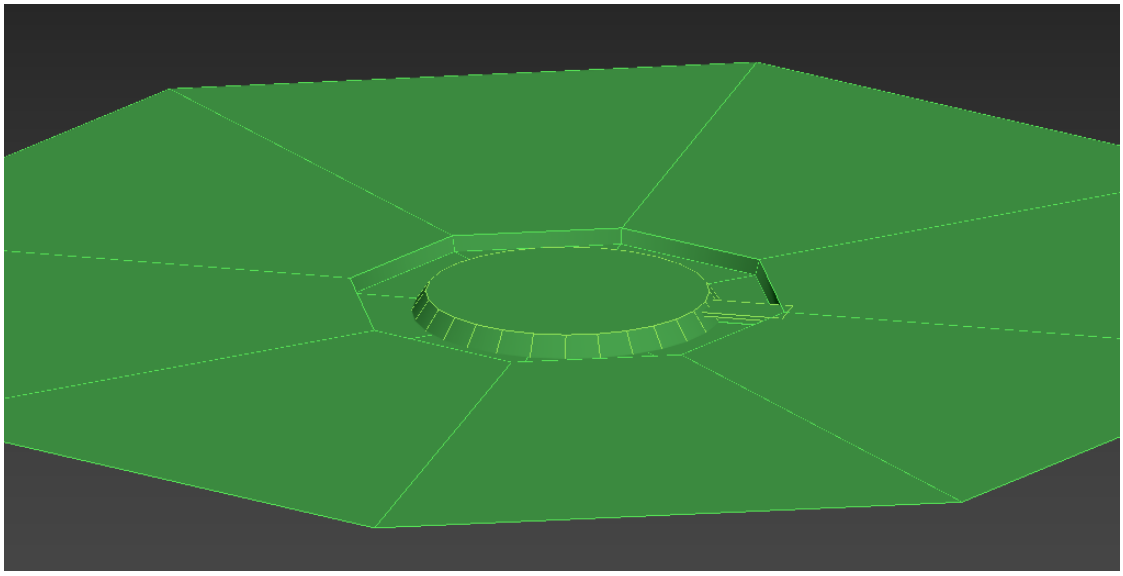
No esta permitido el uso de motores gráficos como Three.js. Todas las primitivas para construir los objetos deben ser implementadas por el alumno.

Menú “rendering”

La idea de este menú es que permita cambiar el modo de rendering entre el default y el modo “normales” en este último todos los pixeles de las superficies deberán pintarse con el valor del vector normal en coordenadas de mundo.
Esto va a ser importante para verificar que las normales estén correctamente definidas.
Se puede implementar de manera sencilla con una variable global que luego se pase como uniform a todos los shaders (en el fragment shader un “if” puede definir que salida usar en función de la variable)

Foso de agua y terreno

Es posible resolverlo mediante superficies de revolución. Pueden ser distintos objetos (centro, puente y terreno)



Iluminación y texturas

No es necesario aplicar iluminación ni texturas, ni colores específicos a los objetos en esta etapa. Es suficiente con que se puedan interpretar correctamente las formas de los modelos

Fecha de entrega: 28 de octubre de 2022

Algoritmos y Funciones requeridos para la implementación

- Curvas de Bezier/Bspline cúbicas: a partir de un arreglo de puntos de control, las funciones deben poder evaluar un punto de la curva en base al parámetro “u”. Además, es necesario que pueda evaluar el vector tangente y normal a la curva

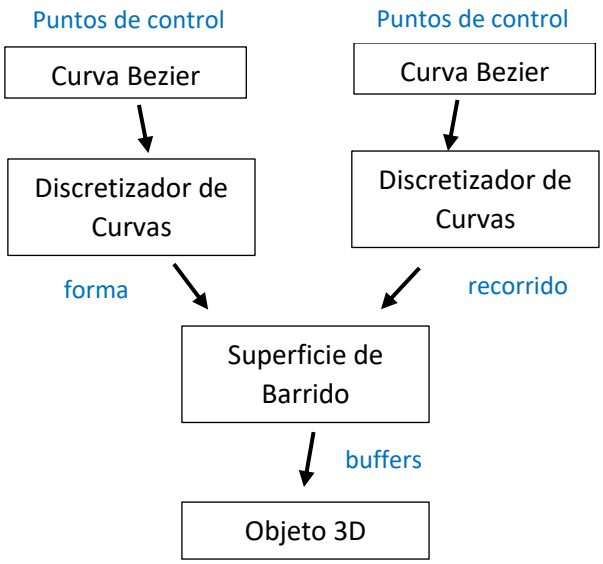
- Discretizador de curvas: dada una curva Bezier o Bspline y un “delta u” devuelve una secuencia de puntos correspondientes a la posición, tangente o normal

- Constructor Objeto 3D: debe ser capaz de crear instancias transformables (posición, traslación y escala) que puedan vincularse jerárquicamente (padre/hijo). Estas pueden tener o no buffers asociados (por ejemplo, en el caso de contenedores)

- Superficie de Barrido: debe ser capaz de recibir como parámetros la forma y el recorrido y devolver los buffers correspondientes a la superficie. Varias piezas del vehículo requieren formas que varían de escala a lo largo del recorrido. La forma más simple de implementar esta capacidad es poder tener múltiples formas ubicadas en distintos puntos del recorrido. La única condición necesaria es que todas las formas posean la misma cantidad de vértices

- Superficie de revolución: idem barrido, pero en lugar de un recorrido, recibe el parámetro “eje”

El siguiente cuadro ejemplifica un caso de uso donde se combinan todos



Ejemplos de objetos que debería poder generarse desde código.