Tarea 3-A: Hungry Bears

Prof: Nancy Hitschfeld K.
Auxiliares: Pablo Pizarro R., Pablo Polanco Galleguillos, Mauricio Araneda H.
Ayudantes: Iván Torres, María José Trujillo Berger

Fecha de entrega: Algún día

Problema

Los osos de Don Pedro se encuentran al otro lado del rio, alimentarlos no es tarea fácil, pues apenas tienen hambre, se enfurecen. Don Pedro ha inventado un simple pero eficaz sistema para enviarles comida sin atravesar el río. La idea es utilizar una honda gigante para lanzarles comida.

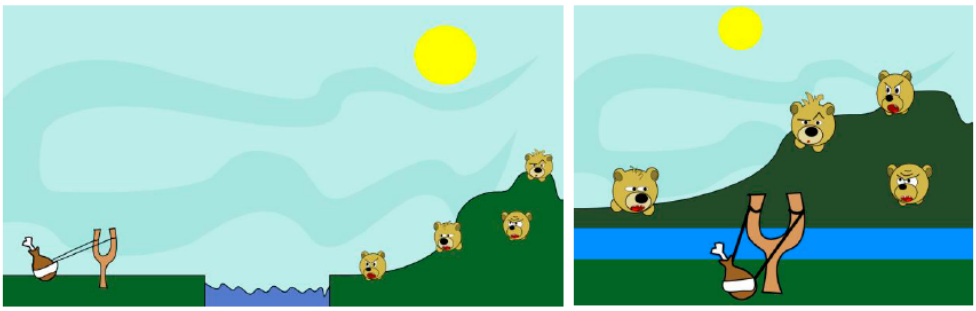

Figura 1: *Imágenes referenciales, pues sus modelos deben ser tridimensionales.

En esta tarea, debe utilizar OpenGL para dibujar. Como lenguaje base se recomienda utilizar Python.

Modelos (4 puntos)

Dibuje 4 osos distintos utilizando OpenGL y distintas estrategias:

- Uno construido en base a figuras GLUT, utilizando al menos unas 4.
- Uno generado a partir de un fichero STL, OBJ u otro formato (Puede encontrar figuras en internet).
- Un tercer oso dibujado por usted en base a especificación manual de vértices y polígonos.

- Para el cuarto oso puede repetir cualquiera de los procedimientos anteriores o utilizar una mezcla de ellos.

Utilizando las estrategias que usted estime conveniente, dibuje 6 alimentos y una honda que permita lanzarlos.

Dibuje un escenario utilizando la(s) estrategia(s) que usted estime conveniente. Deben distinguirse 2 plataformas, un río entre ellas y algún fondo. Formule su diseño para una visualización 3D.

Configure una fuente de luz. Todos sus modelos deben utilizar sombreado, ya sea FLAT o SMOOTH.

Implemente un programa que permita visualizar sus modelos de comida, sus osos y su honda. De forma análoga, implemente otro programa que permita visualizar el escenario y su animación.

El juego (2 puntos)

Para implementar correctamente el juego debe considerar lo siguiente:

- Inicialmente hay un alimento en la honda, y 4 osos al otro lado del rio, en distintas posiciones.
- En cada turno, el jugador podrá lanzar un alimento, si es capturado por el oso, el jugador gana 10 puntos y el oso se retira de la zona. Si el turno anterior también se alimento un oso, el jugador gana 5 puntos adicionales.
- Al final de cada turno debe imprimir en la consola la puntuación actual del jugador.
- El jugador solo dispone de 6 alimentos, si no alcanza a alimentar a todos los osos, pierde la partida.
- El juego puede ser pensado en cualquiera de los siguientes modos:
    - Una visualización 3D en corte transversal: En este caso, el lanzamiento del alimento debe proceder como: Con las flechas es posible estirar el elástico de la honda, lo que le dará mayor intensidad al disparo. Con las flechas es posible rotar la dirección que apunta la honda. Al presionar “espacio”, el alimento es lanzado.
    - Una visualización 3D donde la cámara se encuentre tras la onda, y el alimento pueda ser direccionado libremente. Utilizando las flechas del teclado, fijar la dirección del disparo rotando hacia los lados y hacia arriba y abajo. Con las teclas “a” y “z” es posible controlar la intensidad del disparo (estiramiento del elástico). Al presionar ‘espacio’, el alimento es lanzado.
- Al lanzar el alimento, este debe respetar las leyes de la física, es decir, debe tener movimiento parabólico.
- No se preocupe de la colisión del alimento con el suelo, espere a que el alimento salga del escenario y considere dicho caso como fallido.
- Si el oso atrapa el alimento, ambos desaparecen del escenario. Si no lo atrapa, el oso sigue en juego pero desaparece el alimento.

Bonus Track (máximo 1 punto sobre la nota)

Elija entre los siguientes:

- Día y noche **(1 punto)**: configure dos fuentes de luz, que representarán el sol y la luna (dibújelos utilizando esferas GLUT u otra estrategia). Ambas deben moverse en el cielo alternando entre el día y la noche. El cielo debe cambiar de color en este proceso. Procure que toda la secuencia ocurra en menos de 40 segundos.
- Cámara que sigue al alimento **(0,5 puntos)**: implemente una cámara que se ubique tras el alimento (lo debe seguir durante el lanzamiento). Esta cámara se debe visualizar mientras se mantenga presionado "m".
- Alimento múltiple **(0,5 punto)**: Implemente un alimento que se separe en varios otros cuando el usuario presione la tecla "x".

Presentación-Informe

Se espera que en el informe de esta tarea se explique la metodología llevada a cabo para resolver el problema, la estructura de su código, los resultados y una breve discusión sobre los resultados obtenidos.

En particular especifique los modelos utilizados tanto en el lanzamiento (cálculo de la velocidad inicial), como mientras el alimento se encuentre en el aire.

Observaciones

- **Plazo de Entrega: Algún día. No se aceptan atrasos.**
- No se reciben tareas por otro medio que no sea la sección habilitada en u-cursos.
- Entregue TODOS los archivos que sean necesarios para la correcta ejecución de su programa.
- Puede reenviar su tarea dentro del plazo. En este caso, procure subir nuevamente TODOS sus archivos. El equipo docente sólo tiene acceso a la última entrega que usted realiza.
- La presentación-informe DEBE estar en formato .pdf, no se revisarán otros formatos.
- Las tareas se revisarán utilizando Python 2.7 y sus librerías (instaladores en ucursos).
- Para consultas, se ofrece soporte en el lenguaje Python. También puede utilizar C++ o Java (junto a OpenGL), pero va por su cuenta y debe presentar el código entregado funcionando en su computador.

Hints

- Planifique su tiempo y comience su tarea con anticipación. No comience directamente a programar. Identifique fragmentos de algoritmos, formule clases y objetos útiles, si lo necesita, diseñe máquinas de estados.

- Construya su programa de manera incremental, esto es, vaya respaldado su código de forma que siempre disponga de algo que "funcione".
- Se recomienda utilizar pequeños programas que simplemente dibujen cada una de las partes del juego (objetos, personajes, escenario, etc..). Del mismo modo, puede construir pequeños programas que vayan añadiendo las interacciones entre los distintos actores.
- Una buena metodología es diseñar una clase por cada tipo de actor que interfiera en su programa.