UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Técnicas de Gráficos por Computadora

Enunciado del Trabajo Práctico

Última revisión: 16 de Agosto de 2016

Contenido

Trabajo Práctico

Introducción

La cátedra provee a los alumnos una plataforma para el desarrollo de trabajos prácticos denominada TGC.Core. Esta plataforma es un Framework desarrollado en Microsoft .NET, bajo lenguaje C# y utiliza la API gráfica Microsoft DirectX 9 para el renderizado de gráficos.
Para más información sobre la arquitectura y el uso del Framework dirigirse al documento **"*Guía de TGC framework*"**.
El trabajo práctico de este cuatrimestre consiste en desarrollar un *Ejemplo Creativo* usando el Framework otorgado. En el mismo, el alumno debe integrar todas las técnicas que vio en la cursada, cumpliendo con ciertas restricciones básicas pero aplicando su imaginación para elegir la temática global.
Tendrán que seleccionar el tema del desarrollo de una lista de opciones disponibles, donde cada una presentará los lineamientos y condiciones que deberán ser cumplidos y evaluados.
La cátedra provee un conjunto de ejemplos que son parte del Framework para que sirvan de base al alumno. Estos ejemplos utilizan muchos modelos 3D y texturas que se ubican en la carpeta Media del Framework. El alumno puede hacer uso de todos estos modelos para su ejemplo como así también puede incorporar otros.
Muchas de las técnicas a utilizar son desarrolladas en forma individual en cada uno de los ejemplos del Framework provisto por la cátedra. Es tarea del alumno comprenderlos, dominarlos y lograr su integración para desarrollar su propio ejemplo.

Creación del ejemplo

Deben hacer un fork del proyecto TGC.Group que les servirá como plantilla para desarrollar su propio ejemplo. Una vez realizado el fork deben cambiarle el nombre (TGC.Group) por el nombre de su grupo. De esta manera se podrá identificar el proyecto de cada grupo. El formato especificado por la

cátedra para el nombre del proyecto es: <Año>_<Cuatrimestre>_<Curso>_<NombreDelGrupo>, donde:

- Año deberá tener cuatro dígitos.
- Cuatrimestre deberá ser '1C' o '2C', según corresponda.
- Curso será el código del curso, sin la letra K antecedente.
- NombreDelGrupo deberá tener como máximo 20 caracteres.

P.E. 2016_2C_3072_LosBorbotones

También tendrán que cambiar los valores de Game.settings por los valores reales del grupo.
Pueden encontrar más información en el documento **"*Guía de TGC framework*"**.

Ejemplo Creativo

Para el *Ejemplo Creativo*, los alumnos deberán elegir alguna de las ideas básicas propuestas por la cátedra. Cada una plantea una serie de objetivos y desafíos a cumplir, junto con los requisitos mínimos para poder aprobar el TP. El equipo deberá implementar una única idea, que una vez seleccionada, no podrá ser cambiada.
Cada alternativa también enuncia la funcionalidad obligatoria que debe ser implementada para poder aprobar el trabajo práctico, así como también los ítems opcionales que ayudan a elevar la nota del TP. Además, es posible que el alumno agregue otros ítems bajo su propia iniciativa.

Ítems comunes a todas las ideas

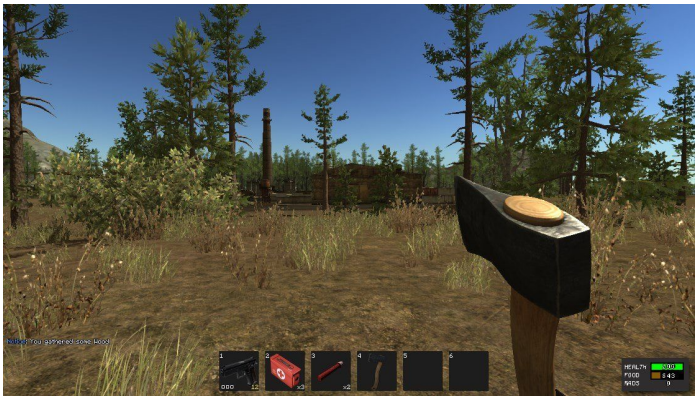
Para cualquier idea que se elija se deberán implementar funcionalidades básicas que son obligatorias:

- **Buena calidad gráfica**: El trabajo práctico es una aplicación gráfica 3D, por lo tanto debe "verse bien". Esto significa que:
  - Debe tener un escenario creado acorde a la necesidad del TP, con la mayor cantidad posible de objetos, distribuidos y ubicados correctamente según la temática elegida. Por ejemplo si se trata de un escenario del tipo outdoor, ubicar una gran cantidad de árboles, plantas, arbustos, pasto, piedras, etc. Si es posible agregar movimientos complejos a dichos objetos, como el efecto del viento sobre el pasto. La calidad gráfica depende del entorno logrado: cuantos más objetos y animaciones se logren, mejor ambientado quedará el escenario. Esto no requiere diseñar nuevos mesh, sino utilizar los que ya están y lograr, mediante técnicas de optimización que se estudian en la materia, que se puedan dibujar la mayor cantidad posible de mesh para dar sensación de escenarios complejos.
  - Todas las texturas del escenario, objetos y personajes deben estar adecuadamente seleccionadas y con la proporción adecuada. Por ejemplo, si estamos modelando el suelo de una cancha de fútbol es necesario utilizar una textura de pasto, pero su apariencia no puede estar toda estirada o deformada por no concordar con la proporción del piso. Tampoco deben tener marcas de agua ni otros defectos que arruinen la ilusión gráfica.
- **HUD**: scoreboards, mapas, previews, indicadores de energía, gadgets, etc. Además dichos textos, menús y controles del TP deben estar correctamente centrados y ubicados en la pantalla, con fuentes y colores que sean compatibles con la temática del TP.
- **Shaders**: El ejemplo debe hacer uso intensivo de los shaders. Esto implica codificar nuevos shaders, utilizar los existentes e integrar las diferentes técnicas que se estudian en la materia, como por ejemplo efectos de post procesado, modelos de iluminación, sombras, efectos de partículas, etc. El shader utilizado debe implementarse en lenguaje HLSL con Shader Model 3.0 (ver **"*Guía de TGC framework*"**).

- **Performance**: el TP debe funcionar al menos a 30 FPS durante toda su ejecución (a excepción de la carga inicial). El alumno deberá hacer principal hincapié en lograr una buena performance. Si un TP cumple toda la funcionalidad pero no alcanza los requisitos mínimos de performance no será aprobado. Los 30 FPS deben lograrse en una computadora con las siguientes características promedio:
    - Notebook de cuatro núcleos de 1.8 ghz.
    - Placa de video Intel integrada con soporte de Shader Model 3.0
    - 4 GB de memoria RAM.
- **Estructura**: El TP deberá contar con una cierta estructura lógica: como mínimo una presentación, luego un menú principal en donde se puedan configurar las distintas opciones, un desarrollo propiamente dicho y un final (se gana, se pierde o se termina el tiempo). Los comandos deben estar disponibles en la pantalla de presentación, o en alguna opción de ayuda. El TP se tiene que poder usar sin leer ningún instructivo externo: simplemente ejecutando el mismo, el usuario debe ser capaz de seguir todos los pasos.
- **Jugabilidad:** Los ejemplos, además de aplicaciones 3D, incluye la noción de ser juegos en sí mismos, por lo que la fluidez de la aplicación y una clara condición de victoria son esenciales para la evaluación del ejemplo de aplicación.
- **Modo God**: para facilitar la presentación es conveniente que se pueda activar un modo "god" o pasar directamente a distintas etapas o niveles del juego.
- **Adaptable:** Teniendo en cuenta que el proyector puede tener distinta resolución (800x600 los más antiguos), el TP tiene que ser "relativamente responsive" y adaptarse a varias resoluciones. Esto quiere decir que no hay que poner posiciones fijas en pixeles que dependan de una resolución particular.

Ideas posibles

Survival Craft – Supervivencia con creaciones

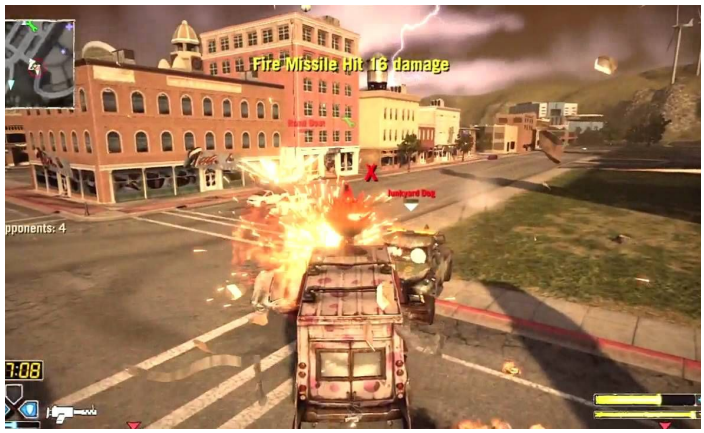


- **Funcionalidades obligatorias para 1ra entrega:**
    - Construir un escenario al aire libre en donde un jugador maneja un personaje en primera persona.
    - El jugador debe poder interactuar con todos los objetos del escenario, obteniendo algo de ellos o destruyéndolos. Objetos tales como árboles, plantas, piedras, agua, etc.
    - El escenario tiene que ser de gran tamaño, con una gran extensión de terreno (debe dar sensación de infinito) El escenario debe contar con al menos 500 objetos.

- El usuario debe tener un inventario en donde los objetos tengan un límite de carga. Los objetos deben poder combinarse con otros objetos del ambiente o con objetos del inventario, de esta forma generar otro objeto a ser utilizado.
- El jugador debe tener un ambiente de supervivencia, debe sufrir cambios de clima (frío, calor) y tener necesidades (sed, hambre, cansancio). Deberá realizar objetos para saciar esas necesidades.

- **Funcionalidades obligatorias para 2da entrega**
  - Los objetos deben tener efectos, tales como viento (árboles, pasto, etc.) o refracciones (ríos, lagos, etc.).
  - El jugador debe tener un menú de inventario, que le permita conocer sus objetos disponibles, pudiendo crear nuevos objetos o descartarlos.
  - El jugador tendrá una descripción por pantalla de su estado actual (frío, calor, sed, hambre, cansancio, normal).
  - El ambiente del escenario debe cambiar aleatoriamente cada cierto tiempo, creando lluvias, clima seco que hagan que el jugador sufra efectos que necesite solucionar. Así como también distintos horarios (día, noche).
- **Funcionalidades opcionales:**
  - Los objetos creados son mostrados en las manos del jugador, teniendo animaciones cuando interactúa con otros objetos.
  - Efectos de humo, fuego, esquirlas, cuando el jugador utiliza objetos.
  - Durante la noche aparición de enemigos que ataquen al jugador.
  - Agregar más estados según situaciones que sufra el jugador (sangrado, enfermedad, etc).
  - Limitar las acciones del jugador según el estado que tenga.

Twisted Metal – Autos chocadores / Derby de demolición



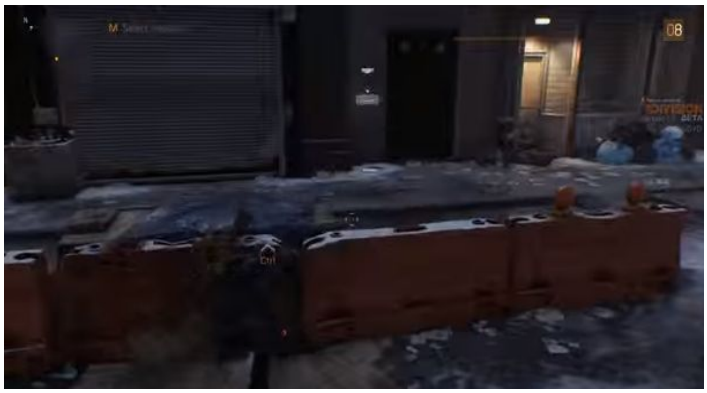
- **Funcionalidades obligatorias para 1ra entrega:**
  - Construir un escenario en donde hay un auto manejado en tercera persona por el usuario dentro de un estadio grande y cerrado. Los autos no pueden salir de este estadio.
  - El auto debe poseer los siguientes movimientos:
    - Acelerar y Frenar.
    - Doblar.
    - Saltar

        - Las ruedas del auto deben rotar acorde al movimiento del auto.
    - El auto debe tener detección de colisiones contra los objetos del escenario (paredes, otros autos, etc). La colisión deberá tener en cuenta los siguientes aspectos:
        - Ser lo más precisa posible.
        - Adaptarse de la mejor manera posible al volumen de la malla del auto.
        - Tener en cuenta la rotación de la malla.
        - No podrá ser utilizada una solución de detección de colisiones simple con AxisAlignedBoundingBox o BoundingSphere (los obstáculos sí pueden ser implementados con AxisAlignedBoundingBox)
        - Al chocar contra un obstáculo, se debe dar una respuesta de colisión apropiada, según la velocidad y ángulo de choque del auto.
    - La cámara debe seguir al coche principal (los laterales no pueden afectar la visión)
- **Funcionalidades obligatorias para la 2da entrega:**
    - Se debe mostrar el tiempo y el marcador por pantalla.
    - El auto debe tener efectos gráficos.
        - Reflejos (Shader de Environment Map).
        - Partículas al chocar.
        - Humo del motor o escape.
        - Deformar la malla del auto al chocar.
    - Poder moverse marcha atrás con el auto y doblar marcha atrás.
    - El estadio debe tener mínimo una luz dinámica de tipo Point Light y generar sombras.
    - Incorporar otro auto con Inteligencia Artificial que persigue al primero e intenta chocarlo.
- **Funcionalidades opcionales:**
    - Agregar otro jugador teniendo un Split Screen.
    - Agregar la posibilidad de disparar y generar daño.
    - Agregar power ups (items) que le den al jugador distintas habilidades temporales:
        - Invencibilidad
        - Posibilidad de atravesar objetos y light paths
        - Turbo
        - Freeze a los contrincantes.
    - Que los autos tengan dos luces dinámicas en sus faroles delanteros y traseros.
    - Efectos de Lens Flare para iluminación del estadio.
- **Bibliografía sugerida:**
    - Libro: Real-Time Collision Detection, autor Christer Ericson, capítulos 4 y 5 sobre OBB.
    - Artículo: http://phors.locost7.info/contents.htm
    - Efecto Motion Blur: http://http.developer.nvidia.com/GPUGems3/gpugems3_ch27.html

Survival Horror – Terror

- **Funcionalidades obligatorias para 1ra entrega:**
  - Construir un escenario cerrado, con muchas habitaciones, pasillos y puertas, donde un jugador maneja un personaje en primera persona.
  - El jugador tendrá linterna, lámpara o vela, cada una de estas iluminarán de forma diferente el escenario.
  - Los objetos de iluminación que use el jugador tendrán una duración, deberá buscar más objetos iguales en el escenario o utilizar otros.
  - Deberán existir diferentes formas de recorrer el escenario, en ciertos lugares un enemigo aparecerá y perseguirá al jugador, teniendo que escapar. El enemigo no puede ser destruido.
- **Funcionalidades obligatorias para 2da entrega**
  - El escenario tendrá muebles, cajas y objetos donde el jugador podrá esconderse si no es perseguido.
  - El enemigo controlado por la IA recorre diferentes caminos, elegidos en el momento de crear el escenario, en caso de ver al jugador según un rango de visión definido, lo perseguirá hasta que el jugador escape o sea atrapado.
  - El jugador, podrá abrir puertas y cerrarlas, cada interacción tomará un tiempo en el cual el jugador no podrá moverse hasta que la puerta esté abierta o cerrada por completo.
  - El enemigo no podrá abrir puertas, salvo que estas estén definidas en su camino de vigilia.
  - Las luces del jugador deben generar sombras.
  - Algunos objetos estáticos del escenario generan luces y sombras.
- **Funcionalidades opcionales:**
  - Los objetos de iluminación serán mostrados en las manos del jugador, teniendo animaciones cuando los mismos pierden su potencia o ya no sirvan.
  - Realizar efectos de niebla, terror cuando el jugador sea perseguido, distorsiones de visión cuando el jugador no tenga objetos de iluminación.
  - Agregar un objeto de iluminación que le de visión nocturna al jugador.

DIVISION - Shooter en 3ra con cobertura



Es un juego de disparos en tercera persona cuya mecánica principal es el sistema de cubrirse con objetos

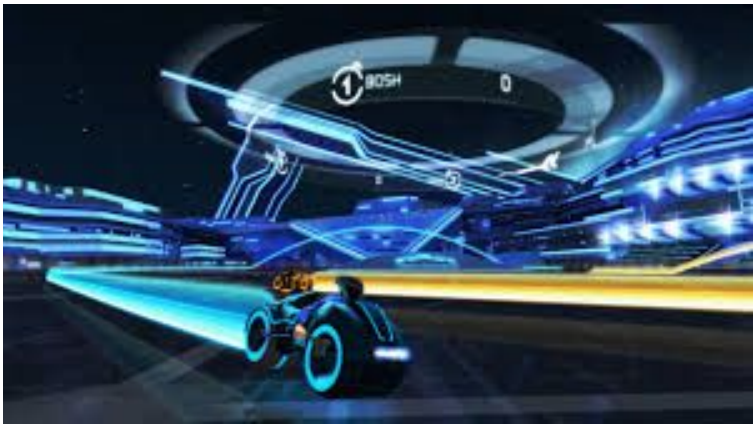
Funcionalidad 1ra entrega:

- Construir un escenario abierto con un jugador principal en 3ra persona que puede realizar disparos.
- Debe haber enemigos que persiguen al jugador y le disparan.
- La inteligencia artificial de los enemigos debe mantener una distancia "segura".
- El escenario debe contar con objetos que permitan al personaje ocultarse de la vista de los enemigos y distinta variedad de objeto visibles.
- El escenario debe contar con al menos 1000 objetos y debe dar las sensación de ser infinito.
- El jugador tiene una barra de energía la cual disminuye cuando un disparo enemigo le acierta.
- El jugador debe poder entrar en un modo de resguardo el cual le permite cubrirse detrás de objetos. Los enemigos no pueden lastimar al jugador cuando se encuentra en este modo.

- **Funcionalidades 2da entrega:**
  - Los enemigos deben poder ocultarse en objetos cercanos al jugador y el jugador no puede lastimarlos cuando se encuentran cubiertos.
  - Los enemigos deben salir del modo de resguardo cuando el jugador está expuesto y tratar de dispararle.
  - El escenario debe tener objetos que pueden explotar dañando a todos los jugadores.
  - Los objetos verdes del escenario (árboles, pasto, plantas) se deben mover simulando el efecto del viento.
  - El escenario debe poseer charcos de agua que reflejan los objetos que tiene alrededor.
  - Se debe aplicar un efecto de partículas cuando los personajes pisan los charcos de agua y cuando explotan los barriles u objetos inflamables.
  - Los objetos deben poseer sombras e iluminación que dan realismo a la escena.
- **Funcionalidad opcionales:**
  - Si el jugador permanece fuera del alcance de los disparos enemigos por un cierto tiempo la energía se recupera.
  - El jugador debe poder correr, caminar y agacharse.

    - El jugador debe poder intercambiar entre un arma principal y una secundaria.
    - Las armas tienen una carga y una cantidad de balas finitas.
    - El jugador puede activar un modo zoom por arriba del hombro en el que la mira se acerca y se puede apuntar mejor.
    - Cuando el jugador hace movimientos fuertes de camara aplicar un efecto de motion blur.
    - El escenario cuenta con cajas de recarga de cartuchos para las armas.
- **Bibliografía:**
    - Tutorial que muestra una posible implementación de Cover Surfaces: https://www.cryengine.com/tutorials/designer-series/ai-entities/ai-cover-surfaces

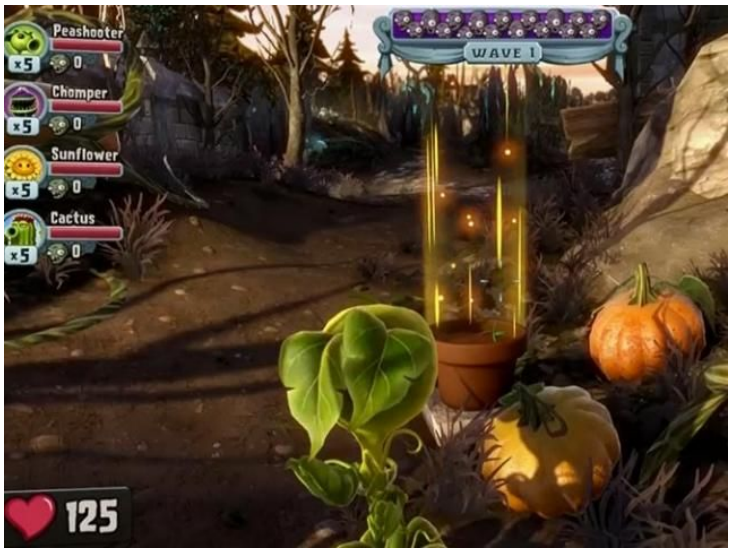
TRON - Grid game



- **Funcionalidad 1ra entrega:**
    - Construir un escenario en forma de grilla plana (the grid) con límites de tribuna alrededor y una ciudad de fondo.
    - La moto del jugador debe poder moverse y controlarse:
        - girando a izquierda y derecha
        - aceleración.
        - freno
    - La moto del contrincante debe manejarse por una IA simple.
    - Ambas motos deben dejar la estela a su paso (light path)
    - El sistema tiene que detectar colisiones entre las motos, el escenario y con el light path generado.
- **Funcionalidades 2da entrega:**
    - Implementar el efecto de bloom o real time glowing (Iluminación de colores brillante) característico de Tron.
    - Agregar un comando para saltar y una mínima física de la moto:
        - al doblar la moto se inclina para un costado y luego se endereza lentamente.
    - Agregar 2 contrincantes adicionales (es decir 4 en total, el jugador y las 3 IA)
    - Agregar obstáculos estáticos al escenario y animaciones varias siguiendo la estética del juego original:
        - carteles luminosos
        - Indicadores de pista o recorrido
        - Bloques luminosos o iluminados

        - Caminos Luminosos
- **Funcionalidad opcionales:**
    - Agregar distintas alturas al escenario, puentes, niveles, rampas.
    - Posibilidad de jugar con otro jugador en split screen (pantalla partida)
    - Agregar proyectiles luminosos que pueden ser lanzados a los contrincantes para retrasarlos.
    - Agregar power ups (items) que le den al jugador distintas habilidades temporales:
        - Invencibilidad
        - Posibilidad de atravesar objetos y light paths
        - Turbo
        - Freeze a los contrincantes.

Plantas Vs. Zombies 3D - Tower Defense



- **Funcionalidad 1ra entrega:**
    - Construir el jardín donde van colocadas las plantas en zonas específicas o macetas con caminos por la que la IA circula pasando por las plantas y los casilleros.
    - Tendrá que haber dos formas de juego, aéreo con selección y 3ra persona donde el personaje recorre el escenario, seleccionando una planta para controlar.
    - Mecánica de soles: Las plantas solo deben poder plantarse si se tiene la cantidad de soles necesaria.
    - Selección de plantas: Se selecciona qué planta se quiere colocar con algún input del usuario.
    - Deben incluirse las siguientes plantas con sus efectos, como mínimo:
        - Repeater Simple: Dispara una arveja.
        - Repeater Doble: Dispara dos arvejas.
        - Sunflower: Periódicamente spawnea un sol que el usuario puede agarrar.
    - Los zombies aparecen en un intervalo de tiempo determinado en la zona del escenario opuesta a la zona de peligro, deben seguir un camino definido. Al

acercarse a una planta, comienzan a comerla y no pueden avanzar hasta haberla devorado.
    - Deben incluirse los siguientes tipos de zombies (El spawn de cada tipo es libre según el grupo, pero todos deben hacer al menos 3 apariciones por nivel):
        - Zombie Normal: Un zombie lento y con hambre de cerebros.
        - Zombie Cabeza de Cono: Zombie más resistente.
        - Zombie Cabeza de Balde: Zombie **mucho** más resistente.
- **Funcionalidades 2da entrega:**
    - Nuevas plantas y zombies:
        - Repeater Congelado: Dispara una arveja que ralentiza zombies.
        - Potato Mine: Mina hecha de papa, explota cuando un zombie la pisa.
        - Jalapeño: Chile picante que elimina toda una línea de zombies, cuando un zombie se acerca
    - Implementar el efecto de fuego para el Jalapeño que elimina una línea de Zombies.
    - Implementar el efecto del zombi “Congelado” y la explosión de la Potato Mine.
    - Implementar el efecto “Transparente” de una planta colocada en el casillero hasta que se haga click en el mismo, confirmandola.
    - Crear una UI para la selección de plantas y una barra de progreso para el nivel.
    - Hordas: Cada un cierto tiempo, los zombies atacan en hordas, llegando muchos más al jardín. A las dos hordas vencidas, deberá ganarse el nivel.
- **Funcionalidad opcionales:**
    - Incorporar efectos en las plantas de movimiento de viento, efectos de partículas en los ataques y efectos cuando son destruidas
    - Implementar el efecto de Iluminación generado por los soles.
    - Aplicar efectos de clima o tiempo que varían durante el tiempo
    - Agregar nuevas plantas o zombies a elección. Algunos ejemplos pueden ser:
        - Catapultas: Disparan en arco.
        - Wallnut: Nuez que sirve como pared, de resistencia superior a las otras plantas
        - Planta Carnívora: Se comen el primer zombie que llega
        - Zombie Michael Jackson: Cada unos segundos, spawnea más zombies con algún efecto especial en la pantalla y una pequeña animación.
        - Zombie con Garrocha: Puede saltar la primera planta que cruza, pero no se salva de las Potato Mines que están enterradas.

Consideraciones adicionales

Guía de temas

Cada idea de trabajo práctico requiere conocer distintos temas que se tratan en la materia. Los temas son tratados en las clases a lo largo de todo el cuatrimestre. El alumno no debe esperar a que llegue la clase de un tema puntual que necesita. Deberá avanzar por su propia cuenta en los temas que deba incursionar para poder desarrollar el TP.
A continuación se brinda un panorama de conceptos generales requeridos en todos los TP junto con la unidad que los explica:

- **Terrenos con Heightmap**: es una forma de crear escenarios de un gran tamaño a partir de una imagen de escala de grises. El concepto es desarrollado en la Unidad 7 “Técnicas de Optimización” en la sección de escenarios “Outdoor”. El framework provee la herramienta “TerrainEditor” que permite generar terrenos con heightmap en forma interactiva.
- **Escenario estático**: un escenario puede construirse mediante la carga de modelos 3D estáticos individuales. El concepto es desarrollado en la Unidad 3 “Conceptos básicos de 3D”. Los modelos estáticos a utilizar pueden ser obtenidos de las siguientes formas:
    - Utilizando los modelos que la cátedra provee junto con el Framework, en la carpeta Media. Los mismos pueden ser cargados mediante la herramienta “TgcSceneLoader” del framework. Su ubicación, rotación y tamaño puede ser configurada en forma manual en el código de la aplicación o también utilizando algunas ejemplos/herramientas como “MeshCreator” y “RoomsEditor”.
    - Creando un escenario completo en la herramienta de diseño 3D Studio MAX y utilizando los plugins de la cátedra para exportar y luego cargar los modelos. Utilizar esta herramienta implica cierto conocimiento de diseño gráfico que no será enseñado en el transcurso de la materia (Ver Guía 3Ds MAX). También pueden aprovecharse modelos ya hechos que el alumno encuentre para 3Ds MAX y luego utilizar los plugins de la cátedra para exportarlos.
    - **Animación**: existen distintas formas de lograr animación, movimiento o una combinación de ambos:
        - Animación esquelética: utilizar la herramienta “TgcSkeletalLoader” para cargar modelos animados mediante la técnica de animación esquelética. El concepto es desarrollado en la Unidad 5 “Animación”. La cátedra provee algunos modelos animados bajo esta técnica. El alumno puede crear nuevos modelos, o agregar más animaciones a modelos existentes mediante la herramienta 3Ds MAX, aunque su utilización puede no resultar trivial. Luego los modelos pueden ser exportados mediante los plugins que provee la cátedra.
        - Animación por KeyFrames: utilizar la herramienta “TgcKeyFrameLoader” para cargar modelos animados mediante la técnica de animación por KeyFrames. El concepto es desarrollado en la Unidad 5 “Animación”. Los resultados obtenidos mediante esta técnica son similares a la animación esquelética, aunque menos flexible y más performantes en algunas situaciones. La cátedra también provee plugins para exportar el contenido de 3Ds MAX.
        - Transformaciones: aplicar transformaciones como traslación, rotación y escalado a objetos estáticos o animados para lograr movimiento. El concepto es desarrollado en la Unidad 2 “Conceptos avanzados de 2D” y luego profundizado en la Unidad 3 “Conceptos básicos de 3D”.
    - **Detección de colisiones**: para detectar si dos objetos chocan entre sí, determinar la selección del mouse o el impacto de un disparo se utiliza el concepto de detección de colisiones. El concepto es desarrollado en la Unidad 6 “Detección de Colisiones”. El Framework brinda herramientas para realizar detección de colisiones con BoundingBox y BoundingSphere, aunque otras estrategias pueden ser implementadas por el alumno.
    - **Shaders**: un shader es un programa escrito en lenguaje HLSL que ejecuta directamente en la GPU. Se puede utilizar para muchas cosas, pero principalmente para lograr efectos especiales, como iluminación, sombras, reflejos, etc. La **“*Guía TGC framework*”** explica cómo utilizar los shaders que ya vienen con el Framework. El tema es tratado en profundidad en la Unidad 8 “Adaptadores de Video” y en el apunte “Guía de Shaders” de esa misma unidad.

Ideas avanzadas

La cátedra busca mejorar en caso de querer incursionar en ideas que poseen una dificultad avanzada. Aquellos grupos que deseen implementar una idea avanzada deberán plantearlo en issues en el repositorio de TGC. Estas ideas tendrán menos soporte de los tutores por tratarse en sí de investigaciones fuera del alcance normal de la materia.
Algunas de ellas son:

- Occlusion culling de una ciudad con muchos edificios.
- Manejar gran cantidad de personajes animados en un mismo escenario, simulación de multitudes.
- Editor de escenarios con geometría constructiva y operaciones booleanas.
- Editor de personajes para poder cambiar aspecto, forma, color, rasgos, etc.
- Editor de animaciones esqueléticas dentro del framework.
- Implementar un Importer del formato Collada para el framework.
- Wrapper de un motor de física como Bullet physics.

También pueden proponerse nuevas ideas que no estén en el listado expuesto anteriormente.

Framework y herramientas de TGC

El objetivo del Framework creado por la cátedra es simplificar el desarrollo del alumno del Trabajo Práctico, haciendo que centre sus esfuerzos en aprender nuevos contenidos de gráficos por computadora, en lugar de tener que lidiar con cuestiones de bajo nivel.
El link a los repositorios público de la cátedra es:

- https://github.com/tgc-utn

Esto puede ser útil para que aquellos que tengan deseo de aprender cómo se construyó puedan hacerlo, y a su vez aportar nuevas ideas de diseño.
También los repositorios en la parte de **Issues** pueden registrar todos los inconvenientes que detectaron o mejoras que crean posibles.

Entrega del TP

Grupos

Los grupos deberán estar formados por 4 integrantes como máximo y 1 como mínimo, sin excepción.
Cada grupo debe tener un nombre que lo identifique.
Los grupos se inscriben a través de un formulario web ubicado en la sección de trabajo práctico del sitio de la materia:

- http://tgcutn.com.ar/trabajos/actual/

Los grupos deberán conformarse para antes del día de Cierre de grupos de TP. Ese día se encuentra especificado en la sección de Cronograma del sitio de la materia.
En el formulario web cargarán los integrantes del grupo y además elegirán la idea de TP a desarrollar.
Aquellos alumnos que no lleguen a conformar un grupo para la fecha del Cierre de grupos de TP deberán realizar el TP en forma individual.
No habrá posibilidad de cambio de grupo luego del Cierre de grupos de TP, bajo ninguna excepción.

Tutores

Cada grupo tendrá asignado un tutor. La interacción entre el tutor y el grupo será vía mail. El tutor contestará preguntas sobre el desarrollo del trabajo práctico. El tutor irá observando el ritmo de evolución del trabajo de cada grupo.

Los grupos tendrán la obligación de enviar un **informe semanal** por mail con el avance del trabajo práctico. El tutor podrá definir nuevos requerimientos y modificaciones de alcance del TP según lo vea conveniente.
El tutor será designado luego de la fecha de Cierre de grupos de TP.

Entregas

El TP posee una entrega de control inicial y dos entregas formales vía email y una presentación oral final.

- Control Inicial: Tendrá el objetivo de tener el escenario armado, sin animaciones y con los modelos seleccionados y ubicados donde corresponda.
- Primer entrega: El avance mostrado debe contener todos los puntos obligatorios indicados en este enunciado para cada idea de TP. En base a la calidad de la entrega y el cumplimiento de los puntos obligatorios el docente fijará una nota. La aprobación de esta instancia es requisito necesario para poder continuar con la cursada de la materia.
- Segunda entrega: cerca del final del cuatrimestre (la fecha está especificada en el cronograma) se debe realizar la 2da entrega. Esta entrega también lleva nota y su aprobación es requisito necesario para la aprobación de la materia.
- Presentación oral: en la última clase del cuatrimestre se hace una puesta en común de todos los trabajos prácticos. Cada grupo muestre el resultado de su TP.

La aprobación del TP es condición necesaria para la aprobación de la cursada de la materia.

Documentación

Para la 2da entrega el desarrollo del TP debe ser acompañado por:

- Video donde se muestra el game play.
- Fotos de los integrantes.
- Tres capturas de pantalla.

Formato de las Entregas

Para realizar una entrega deben crear una release en su repositorio, que debe respetar el siguiente formato “v1.0” donde el primer dígito significa que es la primer entrega y el segundo dígito la cantidad de veces entregada. Por ejemplo, si están reentregando la primer entrega el nombre sería “v1.1”.
Recuerden que en el tag no sólo debe estar el código sino también todos los archivos media.
El nombre del proyecto debe corresponder con el nombre del grupo registrado en el sitio de la materia.
En caso de que el nombre posea espacios, se deberá especificar sin espacios o con guiones bajos.
Ejemplo: “Mi Grupo” puede ser “MiGrupo” o “Mi_Grupo”.
Se deben respetar las convenciones de nombres para los proyectos en .Net framework.
Revisar los Namespace de las clases para evitar problemas.
La entrega del trabajo práctico debe notificarse que está disponible por mail a la casilla de corrección de la cátedra.

Entrega del TP y exposición oral

La notificación de que la entrega está disponible deberá ser enviada por mail a la casilla de corrección de la cátedra:

- **tgc.entregas@gmail.com**

El asunto del mail deberá respetar la siguiente nomenclatura:

- **tgc<b>2016<b>2c<b><curso><b><NombreGrupo>**

<b>: significa espacio en blanco, por favor no escribir literalmente “<b>”
Antes de enviar el mail el alumno deberá revisar meticulosamente que se cumplan todos los “Pasos de deploy del TP” explicados en el documento **“*Guía de TGC framework*”**.
El release debe cumplir a la perfección el formato de entrega especificado anteriormente o será rechazado en forma inmediata.
En caso que algún alumno del grupo haya dejado de cursar deberá ser aclarado en el mail de la entrega del TP.
La entrega debe ser notificada vía mail el día que figura en el cronograma de la materia presentado en la página Web. Ese día se aceptan trabajos prácticos hasta las 23:00 hs. Pasado ese horario no se aceptará ninguna entrega.
Se recomienda prever estos temas con anticipación.
Una vez aprobada la segunda entrega del TP, los alumnos deberán hacer una exposición oral del mismo ante todo el curso. La fecha de exposición oral es posterior a la entrega del TP y también se muestra en el cronograma de la materia de la página Web.
La exposición incluye mostrar funcionando en una computadora con proyector el Ejemplo Creativo desarrollado y explicar a los alumnos las decisiones de diseño y herramientas utilizadas.
El código del TP que será utilizado para la exposición es el de la release aprobada por el tutor. No podrá utilizarse una versión más actualizada.

Nota del TP

La nota del TP se determina en base a los siguientes parámetros:

- Nota de primera y segunda entrega.
- Cumplir con todos los requerimientos obligatorios.
- Requerimientos opcionales agregados.
- Calidad del Ejemplo Creativo entregado.
- Calidad de la exposición oral.
- Entregar cumpliendo el formato de entrega y todas las normas de la cátedra.
- Entregar en tiempo respetando las fecha y horario de entrega.
- La nota disminuye con cada re-entrega que se hace.