Documentación adicional sobre la arquitectura

Nap Time Studios - Holy Spoons

> repo: https://github.com/dimateos/P3

Alumnos

- Baratto, Diego (DiegoBV dbaratto@ucm.es)
- Barea, Juan (JuBarea jubarea@ucm.es)
- Cañellas, Lluis (LluiCS lluiscan@ucm.es)
- Mateos, Diego (dimateos dimateos@ucm.es)
- Rodríguez, Jorge (jorgerodrigar jorger09@ucm.es)
- Sanz, Gonzalo (gonzsa04 gonzsa04@ucm.es)

General

> Como ya hemos ido explicando en las presentaciones, resumidamente:

Tenemos un Game que inicia los sistemas necesarios (physicsManager_, renderManager_, soundManager_, messageSystem_...) y ejecuta el el bucle principal.

- stepPhysics (actualiza phsx)
- updateNodes (activa flags en los objetos de render que necesitan actualizar su posicion)
- handleEvents (input)
- update (dentro tiene late_update)
- renderFrame (ogre renderiza un solo frame)
- updateSound (actualiza posiciones de sonido 3D)

> la actualizacion de las posciones de los nodos de render ocurre en late_update del componente de render

Tenemos una GameStateMachine para manejar los GameStates

- Con su lista de GO y cada uno de ellos con su lista de Comps

Los GO tiene posicion y orientacion propia. Tanto GO como Comp son

- `Activable` (setActive, toggle, etc)
- `identifiables` (tienen id, grupo, etc)
- `Initiable` (no se inician al contruir sino al añadir a la escena activa, etc)

> se puede poner posicion / ori a cualquier GO y este sobreescribe a phsx y a ogre

Cosas bastante mejorables / mal / no dio tiempo

Teniamos pensado solucionar unas cuantas pero al final preferimos meter un poco mas de contenido y assets al juego.

Proyectos

- Separar los archivos en carpetas fisicas y no solo filtros
- Eliminar los restos de memoria, no podiamos usar checkML.h eficientemente
    - Y cuando empezamos a usarlo ya era muy dificil encontrar las fugas
        - Hay muchas pero lo probable es que sea algo de GO / Comps que no hemos visto y que produce leak por cada uno, etc..
    - Al incluirlo en alguno proyectos sus #define producian errores
        - En tiempo de ejecucion en SDL / ogre
        - En tiempo de compilacion en Phsx

Arquitectura

- Dependemos de la libreria de json como estructura que transmite las configuraciones de los GO y componentes, y en mas sitios.
    - Esto era tanto esfuerzo cambiarlo que no ibamos a tocarlo
- Despues del hito 2 hicimos la clase `globalConfig` que lee de el archivo `user/config.json` y `globalCFG.json` y guarda valores generales de la aplicacion en mapas.
    - Deberiamos haber recogido todos los valores constantes y #define que aun hay todavia por el codigo y reunirlos ahi.

GameObjects

- Los componentes arrastran un parametro extra en sus `handleEvents`, `update`, `late_update`
    - `virtual void update(GameObject* o, double time);`
    - Es una referencia al gameObject dueño del componente
        - Posteriormente añadimos `owner_` a los componentes y esto ya no lo necesitan
        - Totalmente trivial de quitar pero no encontramos el momento
- Las escalas de los objetos no estan unificadas
    - Querriamos haber podido especificar una escala del GO y que los componentes de render y phys se ajustasen a esta.
    - Como usamos .mesh en ogre no estan establecidos sus tamaños y pueden ser cualquiera
    - Por ello pretendiamos calcularlo tras cargar y ajustar su escala para que su tamaño coincidiese con el GO y hitbox.
    - Al principio teniamos algo funcionando pero en realidad no le puedes preguntar a ogre el tamaño de la hitbox hasta que no ha renderizado una vez el objeto etc...
        - Lo conseguimos solucionar pero ya teniamos varios niveles configurados, etc y no preferimos no perder el tiempo en retocar todos los jsons
- Ahora mismo el cambio de estado (menu -> juego) se notifica a los componentes que lo necesitan por mensaje.
    - Seria mejor que fuese un método virtual que usase polimorfia
    - Cambiarlo es trivial pero no encontramos momento

SceneLoader, GoFactory

- La clase SceneReader depende de la libreria de json y podria estar separado facilmente
    - Las escenas deberian de precargar las listas de GoStructs de cada escena
- La clase GoFactory junta la factoria de gameobjects y componentes
    - Deberian estar separados
    - Se podria implementar un sistema parecido al mapa de los comps para los prefabs
    - El mapa de prefabs ahora mismo lo aloja el SceneReader junto con las escenas

Features

- Conseguimos meter animaciones con esqueletos en ogre pero originaban problemas con las texturas, y ya decidimos que no merecia la pena adaptarlas todas.

- Estabamos trabajando en un menu in-game de opciones pero como su funcionalidad era simplemente era la misma que editar `user/config.json` lo dejamos para el final y no lo hemos hecho.

- No se puede editar el volumen general del juego in-game aunque los metodos para hacerlo si que estan. Como queriamos meterlo en el menú de opcines tampoco estña al final.