Informática Gráfica II

Object-oriented Graphics Rendering Engine

Descripción del esqueleto del proyecto

Material original: Ana Gil Luezas
Adaptación al curso 24/25: Alberto Núñez
Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

Estructura del proyecto

- El proyecto está formado por dos clases
  - IG2App
  - IG2ApplicationContext
- IG2App
  - Contiene la lógica de nuestra aplicación
- IG2ApplicationContext
  - Crea Root
  - Crea el Scene Manager
  - Crea el Overlay System
  - Otras tareas de configuración…

Clase IG2App

```
class IG2App : public  OgreBites::IG2ApplicationContext, OgreBites::InputListener{

public:
    explicit IG2App() : IG2ApplicationContext("IG2App") { };
    virtual ~IG2App() { };

protected:
    virtual bool keyPressed(const OgreBites::KeyboardEvent& evt);
    virtual void setup();
    virtual void setupScene();
    virtual void shutdown();


    Ogre::SceneNode* mSinbadNode = nullptr;

    Ogre::SceneManager* mSM = nullptr;
    OgreBites::TrayManager* mTrayMgr = nullptr;

    Ogre::Light* light = nullptr;
    Ogre::SceneNode* mLightParent = nullptr;
    Ogre::SceneNode* mLightNode = nullptr;

    Ogre::SceneNode* mCamNode = nullptr;
    OgreBites::CameraMan* mCamMgr = nullptr;
};
```

Clase IG2App

```
class IG2App : public  OgreBites::IG2ApplicationContext, OgreBites::InputListener{


public:
    explicit IG2App() : IG2ApplicationContext("IG2App") { };
    virtual ~IG2App() { };

protected:
    virtual bool keyPressed(const OgreBites::KeyboardEvent& evt);

    virtual void setup();


    virtual void setupScene();

    virtual void shutdown();


    Ogre::SceneNode* mSinbadNode = nullptr;

    Ogre::SceneManager* mSM = nullptr;
    OgreBites::TrayManager* mTrayMgr = nullptr;

    Ogre::Light* light = nullptr;
    Ogre::SceneNode* mLightParent = nullptr;
    Ogre::SceneNode* mLightNode = nullptr;

    Ogre::SceneNode* mCamNode = nullptr;
    OgreBites::CameraMan* mCamMgr = nullptr;
};
```

Es subclase de IG2ApplicationContext

Eventos (presionar tecla)

Configuración del sistema

Configuración de la escena

Finalización de la aplicación

Clase IG2App

```
class IG2App : public  OgreBites::IG2ApplicationContext, OgreBites::InputListener{


public:
    explicit IG2App() : IG2ApplicationContext("IG2App") { };
    virtual ~IG2App() { };

protected:
    virtual bool keyPressed(const OgreBites::KeyboardEvent& evt);
    virtual void setup();
    virtual void setupScene();
    virtual void shutdown();


    Ogre::SceneNode* mSinbadNode = nullptr;

    Ogre::SceneManager* mSM = nullptr;

    OgreBites::TrayManager* mTrayMgr = nullptr;

    Ogre::Light* light = nullptr;
    Ogre::SceneNode* mLightParent = nullptr;
    Ogre::SceneNode* mLightNode = nullptr;

    Ogre::SceneNode* mCamNode = nullptr;
    OgreBites::CameraMan* mCamMgr = nullptr;
};
```

Elemento de la escena (el ogro)

Instancia de SceneManager

Instancia de TrayManager

Una luz

Nodo asociado a una cámara

Main

```
#include "IG2App.h"


int main(int argc, char *argv[]){
    IG2App app;
  try {
    srand((unsigned)time(NULL));
    app.initApp();
    app.getRoot()->startRendering();
  }
  catch (Ogre::Exception& e) {
    Ogre::LogManager::getSingleton().logMessage(”Exception has occured: "
                                                + e.getFullDescription() + "\n");
  }
  app.closeApp();
  return 0;
}
```

IG2ApplicationContext::initApp()

```
void IG2ApplicationContext::initApp(){
    createRoot();
    if (oneTimeConfig())
        setup();
}
```

Invoca a IG2App::setup()

```
void IG2ApplicationContext::createRoot(){
    Ogre::String pluginsPath;
    pluginsPath = mFSLayer->getConfigFilePath("plugins.cfg");

    if (!Ogre::FileSystemLayer::fileExists(pluginsPath)){
        OGRE_EXCEPT(Ogre::Exception::ERR_FILE_NOT_FOUND, "plugins.cfg", "IG2ApplicationContext::createRoot");
    }
    mSolutionPath = pluginsPath;
    mSolutionPath.erase(mSolutionPath.find_last_of("\\") + 1, mSolutionPath.size() - 1);
    mFSLayer->setHomePath(mSolutionPath);   //Config files in /bin
    mSolutionPath.erase(mSolutionPath.find_last_of("\\") + 1, mSolutionPath.size() - 1);

    mRoot = new Ogre::Root(pluginsPath, mFSLayer->getWritablePath("ogre.cfg"),
                                        mFSLayer->getWritablePath ("ogre.log"));

    mOverlaySystem = new Ogre::OverlaySystem();
}
```

Configuración del las rutas para los ficheros de configuración

Creación de Root

Creación del OverlaySystem

Estructura del proyecto

- Algunos de los atributos de IG2ApplicationContext
  - Desde IG2App accedemos por herencia

```
Ogre::Root* mRoot;          // Instancia de Root
NativeWindowPair mWindow; // La ventana


Ogre::FileSystemLayer* mFSLayer;      // Abstracción del Sistema de ficheros
Ogre::OverlaySystem* mOverlaySystem;  // Sistema Overlay

Ogre::String mSolutionPath;         // Rutas relativas al directorio de la solución

Ogre::RTShader::ShaderGenerator * mShaderGenerator; // Instancia del generador de shaders
```

IG2App::setup()

```
void IG2App::setup(void){

    IG2ApplicationContext::setup();

    mSM = mRoot->createSceneManager();

    mShaderGenerator->addSceneManager(mSM);

    mSM->addRenderQueueListener(mOverlaySystem);
    mTrayMgr = new OgreBites::TrayManager("TrayGUISystem", mWindow.render);
    mTrayMgr->showFrameStats(OgreBites::TL_BOTTOMLEFT);
    addInputListener(mTrayMgr);

     addInputListener(this);

     setupScene();
}
```

Invoca setup() de la clase padre

Crea el scene manager

Registra el scene manager en el RTSS

Configuración del overlay system

Añade el objeto en el Listener

Invoca **setupScene()**

IG2ApplicationContext::setup()

```
void IG2ApplicationContext::setup(){

    mRoot->initialise(false);
    createWindow(mAppName);
    setWindowGrab(false);

    locateResources();
    initialiseRTShaderSystem();
    loadResources();

    mRoot->addFrameListener(this);
}
```

Inicializa Root

No crea ventana automáticamente

El ratón no permanece en la ventana

Localiza y carga los recursos

Añade la clase como listener

Clase IG2ApplicationContext

Hereda de **FrameListener** (observador de root) y mantiene una lista de observadores de eventos de entrada (**InputListener ***)

```
class IG2ApplicationContext : public FrameListener  {

  public:
     virtual void createRoot();
     virtual bool frameStarted(const Ogre::FrameEvent& evt) {
                                  pollEvents(); return true; }
     void pollEvents();    // Avisa a los observadores

  protected:
     Ogre::Root* mRoot;
     NativeWindowPair mWindow;
     Ogre::FileSystemLayer* mFSLayer;
     Ogre::OverlaySystem* mOverlaySystem;
     Ogre::RTShader::ShaderGenerator * mShaderGenerator;
     …
     std::set<OgreBites::InputListener*> mInputListeners;
};
```