# Wczytywanie plików konfiguracyjnych (biblioteka)

TinyXML 2 jest prostym, małym, wydajnym, parserem plików xml, który łatwo się integruje w aplikacje. TinyXML 1 używał pamięć nie efektywnie i nie sprawdzał się dobrze dla urządzeń mobilnych

Oba parsery posiadają: podobne API, są oparte o DOM i wspierają UTF-8.

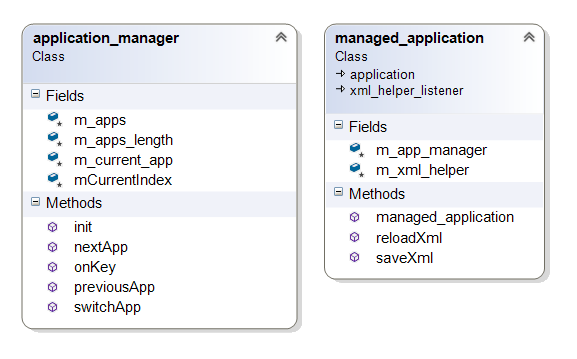
Natomiast TinyXML 2 alokuję mniej pamięci (około 40% co TinyXML 1), odczytuję dane 5 razy szybciej niż jego poprzednik, zawiera nowocześniejsze wykorzystanie c++ (przestrzenie nazw), oraz właściwą obsługa białych znaków

TinyXML 1 może raportować błędy parsowania oraz jest dobrze przetestowany I sprawdzony.

# Wstępna integracja

## Wstępna integracja przykładów

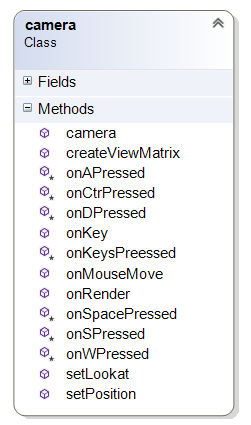
Polegała na stworzeniu klasy zawierającej przykłady.



Klasa application\_manager zawierała obsługę interakcji z użytkownikiem, którą przekazywała do obiektu klasy managed\_application która zawierała przykład z SB6.

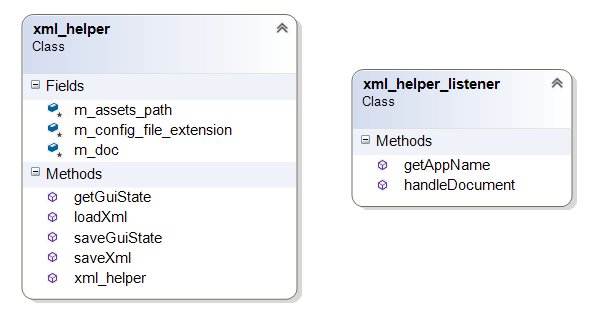
Klasa managed\_application obsługiwała wczytywanie danych konfiguracyjnych, renderowanie przykładów oraz przekształcanie sceny takie jak rotacja, translacja, skalowanie.

## Wstępna obsługa kamery



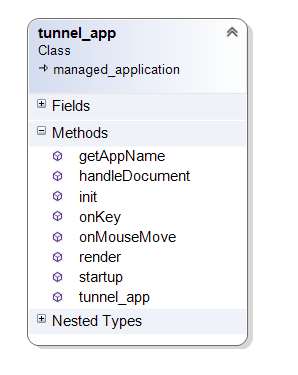
Klasa camera umożliwiała obsługę poruszania się po scenie za pomocą klawiszy WSAD, spacji (zwiększenie wysokości), control (obniżenie wysokości), shift (zwiększenie czułości) oraz myszki do obrotu sceny. Tworzyła ona także macierz widoku.

## Wstępna obsługa ładowania plików konfiguracyjnych do przykładów

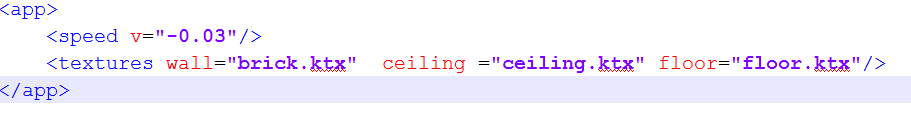


Klasa xml\_helper obsługiwała ładowanie plików XML z dysku, współpracuję ona z xml\_helper\_listener do zwrotnego przekazania załadowanego pliku do managed\_application w celu inicjalizacji przykładu.

## Przykład integracji przykładu do jednej aplikacji

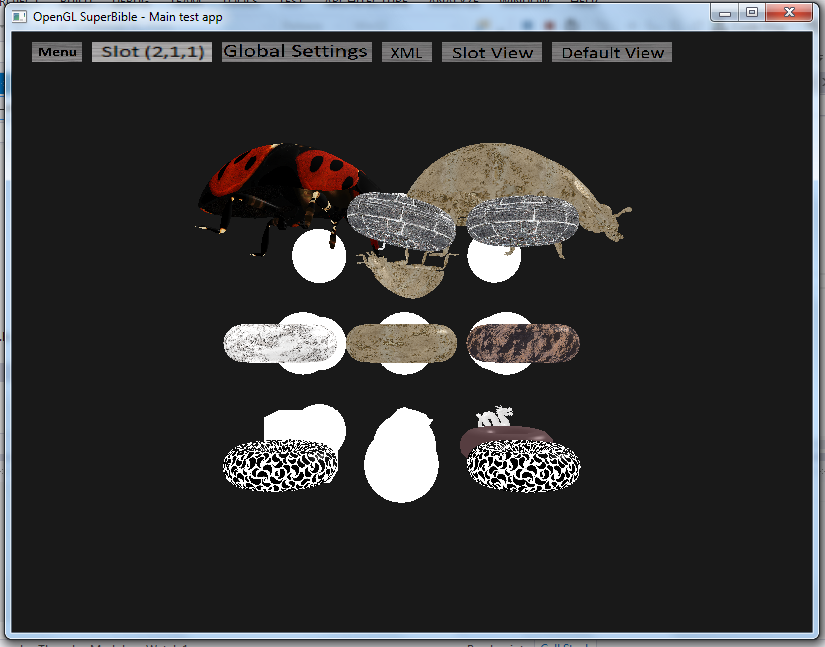


Jest to przykład z SB6 włączony do aplikacji. Metoda getAppName() jest dla znajdowania właściwego pliku konfiguracyjnego, a onMouseMove() jest wykorzystywane przez klasę camera.



Jest to plik konfiguracyjny dla klasy tunel\_app.

# Końcowa aplikacja



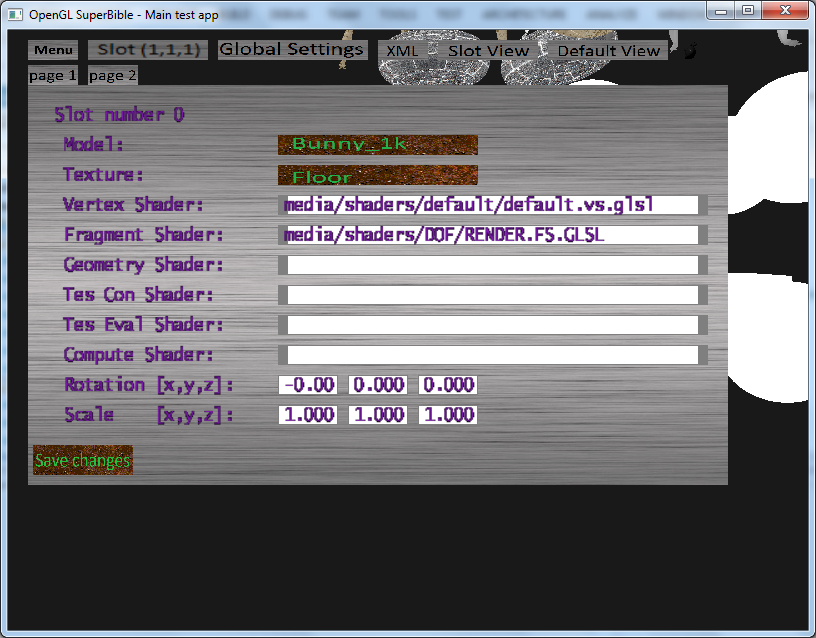
Wygląd aplikacji

Aplikacja składa się z 27 odzielnych obiektów ułożonych w kostkę 3x3x3. Każdy z obiektów ma przypisane indywidualne właściwości, które możemy, niezależnie od innych obiektów, zmieniać. Są to:

* Model,
* Tekstura,
* Wszystkie typy shaderów,
* Rotacja w trzech osiach,
* Skalowanie w trzech osiach.

Dodatkowo, dostępne są ustawienia globalne – dla całej sceny, zawierające:

* Pozycję światła na scenie,
* Albedo materiałów dla światła rozproszonego oraz ‘lustrzanego’,
* Siłę światła lustrzanego,
* Komentarz odnośnie sceny.



Parametry pojedynczego obiektu

Wszystkie te parametry, wraz z obecną pozycją kamery, możemy zapisać do wybranego pliku XML, a następnie wczytać taką konfigurację w dowolnym momecie.

Plik XML jest zorganizowany w następujący sposób:

<app>

<light posX="98" posY="100" posZ="100"/>

<material\_properties diffuse\_albedoX="0.1" diffuse\_albedoY="0.1" diffuse\_albedoZ="0.1" specular\_albedo="0.3" specular\_power="100.5"/>

<camera posX="-10" posY="0" posZ="-4" eyeX="1" eyeY="-0.2666668" eyeZ="0" centerX="0" centerY="0" centerZ="0" upX="0" upY="1" upZ="0" hAngle="766.76941"/>

<info comment="FIRST COMMENT"/>

<objects>

<object coordX="0" coordY="0" coordZ="0" model="media/objects/bunny\_1k.sbm" texture="media/textures/floor.ktx" rotX="-0" rotY="0" rotZ="0" scaleX="1" scaleY="1" scaleZ="1" vsPath="media/shaders/default/default.vs.glsl" fsPath="media/shaders/DOF/RENDER.FS.GLSL" gsPath="" tesPath="" tcsPath="" csPath=""/>  
 <object coordX="1" (...)  
 </objects>

</app>

## Architektura

Na rdzeń aplikacji, wyłączając GUI, składają się dwie główne klasy: MainApp i SceneObject, reprezentujące odpowiednio aplikację i indywidualny obiekt na scenie, struktura SceneObjectParams przetrzymująca parametry każdego z obiektów oraz wcześniej wspomniana pomocnicza klasa xml\_helper, ułatwiająca obsługę plików XML.

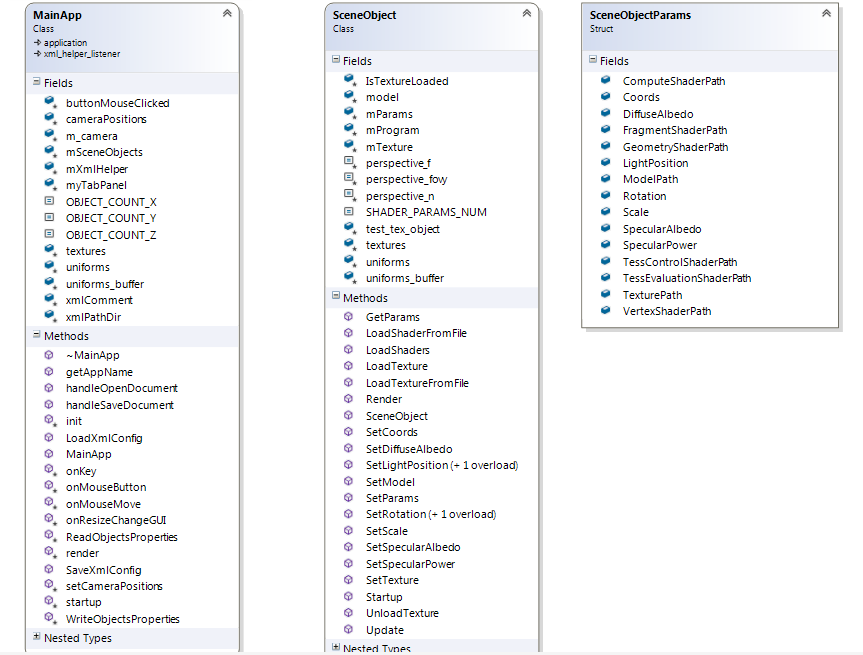


Diagram użytych klas

// TODO adatczuk