ETGG3802

Lecture5: SceneManagement

Unity / Unreal / etc.

- GUI Scene Editor built into Engine
- Is it a necessary part of a game engine?
  - Consider:
    - C#/Monogame (Stardew Valley, Axiom Verge, Celeste, etc.)
    - Many in-house / custom engines
- For us:
  - (Jason will…) use **blender** and an **ogre exporter** script to create an **xml scene** file
    - Parent / Child
    - Meshes / Camera / Lights are exported
    - Scripts, tags, etc. can be linked by adding a "property"
  - Look at the sample scene (space invaders)

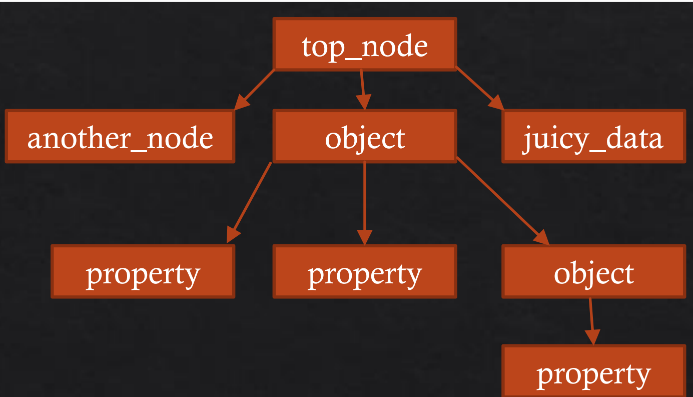
XML structure

- Terminology:
  - top_node, another_node, object are all “tags”
  - img, x, name are all “attributes”
  - the first line is the “XML declaration” – it’s not considered a node.
  - all tags have an opening and closing tag (indicated with a slash).
    - Some tags are self-closing (have a slash at the end)
  - If there is anything outside <>, it is the “text” that goes with the enclosing tag.
- White-space, tabbing are irrelevant
  - There are tools to “prettify” it
- <!--   and --> are block comments
- Should be thought of as a tree

```
<?xml version="1.0" encoding="UTF-8"?>
<top_node img=“bunny.jpg”>
     <another_node x=“14”, y=“20”/>
     <object name=“foo”>
          <property name=“age” value=“55”/>
          <property name=“style” value=“smooth”/>
          <object name=“baby-foo”>
               <property name=“style” value=“sleepy”/>
          </object>
     </object>
     <juicy_data>
          This is the text that goes with juicy
     </juicy_data>

</top_node>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<top_node img=“bunny.jpg”><another_node x=“14”,
y=“20”/><object name=“foo”><property name=“age”
value=“55”/><property name=“style”
value=“smooth”/><object name=“baby-foo”><property
name=“style”
value=“sleepy”/></object></object><juicy_data>This is
the text that goes with juicy</juicy_data></top_node>
```



Parsing Strategy

- Build off the start from Lab5
- If you encounter a “node” – start creating a game object
  - pick off *immediate* children for
    - creation-attributes: tag, script (next lab) [needed before creating]
    - position, scale, rotation [used after creating]
    - whether or not to create a component
      - light, camera, etc.
  - Ignore other nodes (some we’ll have already visited, like position)
- Parent-Child relationship?
  - Store current parent as part of the recursion?
  - Pass NULL initially?