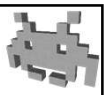
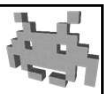
Course Plan



lec. 1: **Introduction**
lec. 2: **Mathematics** for 3D Games
lec. 3: **Scene Graph**
lec. 4: Game **3D Physics** + **Particle Systems**
lec. 5: Game **Particle Systems**
lec. 6: Game **3D Models**
lec. 7: Game **Textures**
lec. 8: Game **3D Animations**
lec. 9: Game **3D Audio**
lec. 10: **Networking** for 3D Games
lec. 11: **Artificial Intelligence** for 3D Games
lec. 12: Game **3D Rendering Techniques**

23

Reminder: inverse of a composite transform (or, in general, function)



$$(T_B \circ T_A)^{-1} = T_A^{-1} \circ T_B^{-1}$$

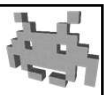
- The inverse of "first Ta then Tb" is "the inverse of Tb" followed by "the inverse of Ta"
- As it's natural! If you…
  - "take a step *forward*, *then,* turn by 90° *clockwise*"

…then, to go back to the starting pos, you need to…

  - "turn by 90° *counter-clockwise*, *then*, take a step *backward*"

24

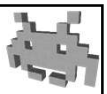
The camera in the scene graph



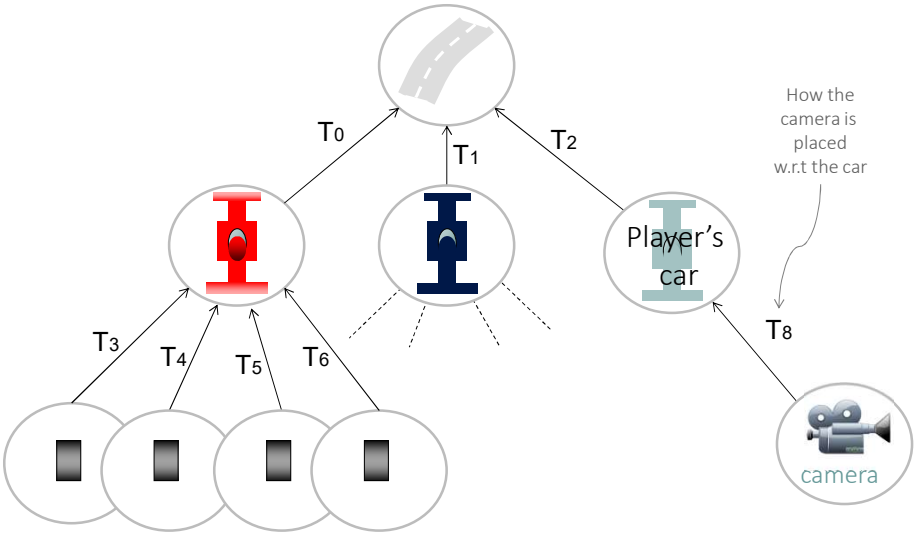
- Camera:
  - Like any other object in the scene, the camera sits in a node the scene-graph
  - for the scene to be rendered, there must be a camera somewhere in the graph!
  - View Space = Local Space of the camera
  - (Screen Space is a similar, and sometimes equivalent, concept)
- the View Space is crucial for the rendering engine
  - In view space, coordinates describe where things are in front of the camera!
  - For example: $z > 0 \Rightarrow$ in front of the camera, $z<0 \Rightarrow$ behind the camera (don't render)
- Camera animations = move camera
  - by anything that changes its global transformation
  - e.g. a script changing its local transform, or the one of it's parent

25

The camera is in the scene graph

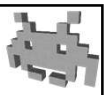


E.g.: to make the camera follow the car…



26

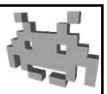
Transforms for the Graphics engine (link to Computer Graphics course)

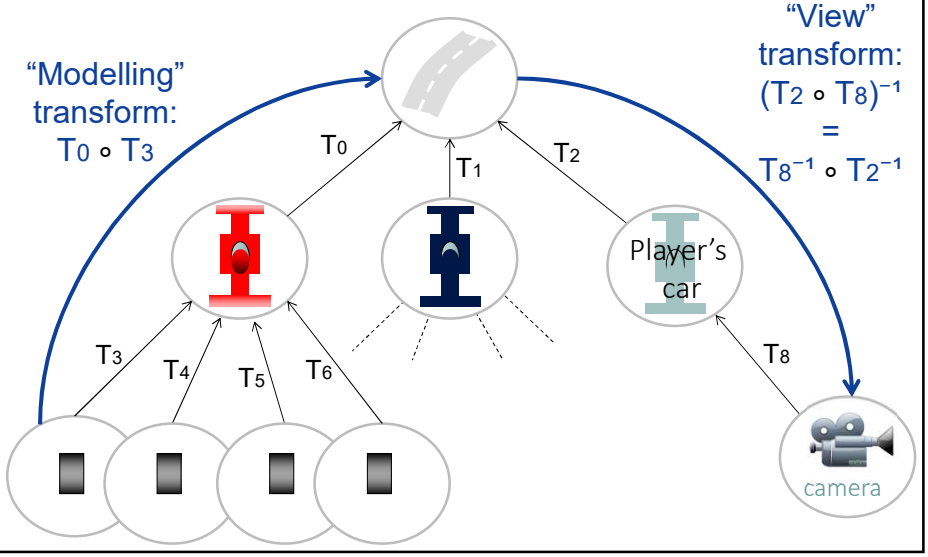


- The rendering engine uses a few standard transformations, when rendering an object,
- They are named:
  - "Model" matrix: from object space to world space
    - Captures how the scene is **modelled** (by a scener)
  - "View" matrix: from world space to view space
    - Captures how the scene is **viewed** (by the camera)
  - "Model-View" matrix: from object space to view space
    - ("matrix" only because trasnforms are usually modelled as 4x4 matrices by Rendering engines & graphics APIs)
- Computing them from the scene graph is easy

27

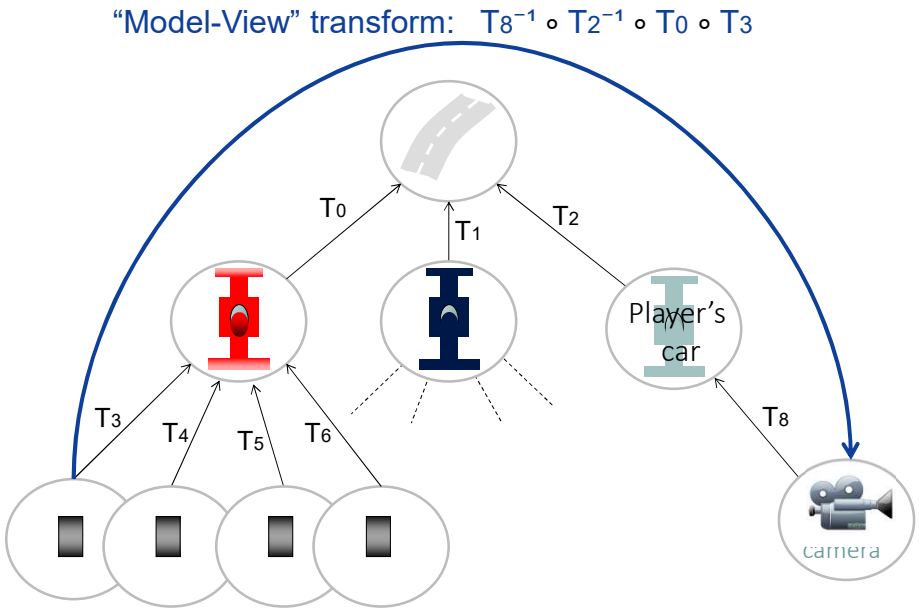
Transforms for the Graphics engine





28

Transforms for the Graphics engine



29

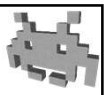
Authoring a 3D scene in a game

- E.g. as a part of the Level Design
- Two different parts, by different artists:
  - 3D modellers make «scene props»
    - the 3D models to be assembled
    - (including their texutres etc)
  - sceners compose the scene
    - they assemble the props into a Scene Graph

= asset

33

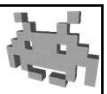
Authoring a 3D scene



- Examples of other assets associated to a scene
  - a Collision Mesh (a Geometry Proxy)
    - one for each "solid" scene-prop
    - can this be made automatic? Possible, not easy
    - assigned to nodes (for dynamic objects), or (for static objects) possibly all merged into one
    - needed for: physics, visibility computation, AI, plus all sorts of gameplay reasons…
  - a Navigation Mesh (aka AI mesh)
    - usually, one for the entire scene (stored in the root node)
    - needed by: AI (routing – see later)
    - can this be made automatic? Possible, not tribial

34

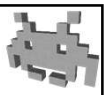
Authoring a 3D scene



- Examples of other assets associated to a scene:
  - Scripts
    - by the level designer
  - Sky box
  - Outer terrain mesh…
  - Ambient sounds
  - Other data such as spawn points, and more

35

Scene Graph as a data structure



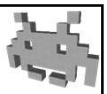
- Each engine / library adopts its own solution
- No standards
  - but file formats exists which can include a scene graph: e.g. COLLADA

Typical concepts:

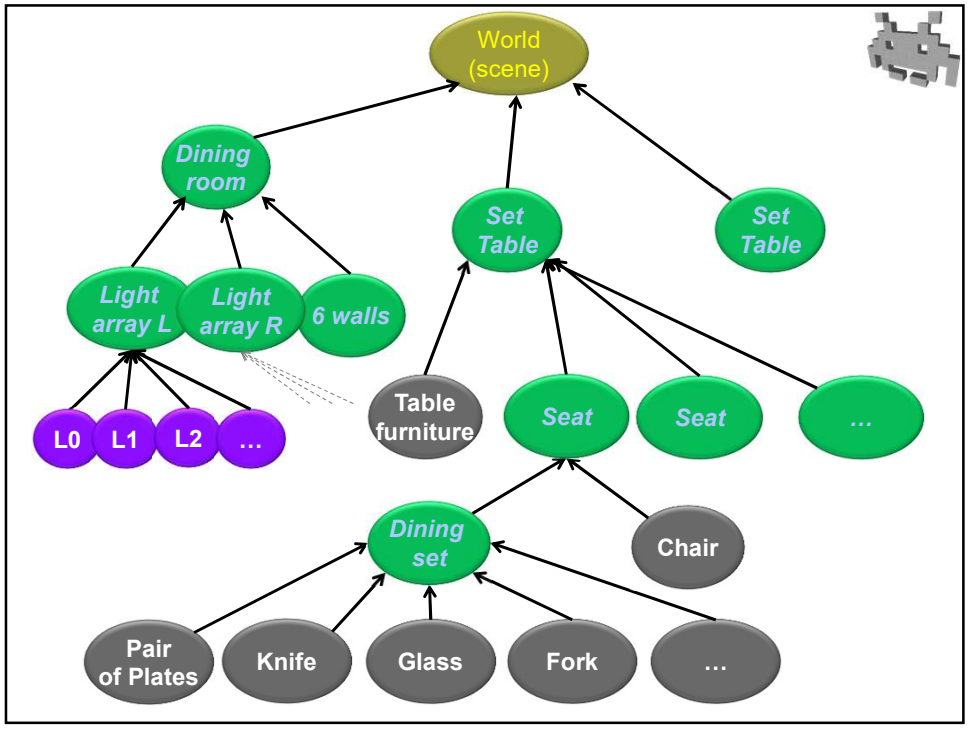
- each Node class stores
  - the local transform
  - link to parent
  - maybe, and/or to children, sibilings…)
  - links to instances / assets
- global transforms / inverse are computed on demand
- some mechanism is used for repeated sub-trees

36

Example: a dining table



37

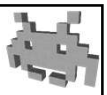


38

Example: a dining table

39

Nodes of a scene-graph in unity
GameObjects & Transforms

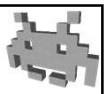


A node = a GameObject with

- a transform field, containing
  - its local transform
  - links to Parent, Children (and siblings) – which are transforms
- any number of associated "components", which represent anything residing in that node, like
  - Meshes (to display at this nodes)
  - Cameras: active one(s) produces the rendering(s)
  - "RigidBodies": objects controlled by the physics
  - "Colliders": geom proxies used for collisions
  - "Particle systems" : (i.e. the "emitters" of particles)
  - Sound producers / receivers
  - Scripts …
  - basically any asset!

40

Nodes of a scene-graph in unity
GameObjects & Transforms

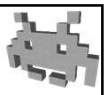


- The Transformation actually stores the local transf:
  - **`localPosition`**, **`localRotation`**, **`localScale`**
  - goes from a node to its parent
- the Global transformation can be accessed via the properties:
  - **`position`**, **`rotation`**, **`scale`** ("global" is left implicit)
  - what does getting / setting them really do? (exercise)
  - this it doesn't always work for "scale":
    ~~**`scale`**~~ `lossyScale` (read only)
    Why? (A: it's because anisotropy)

feels like assigning / reading a field, actually means invoking setters/getters (C# trick)

42

Digression on Unity properties and components

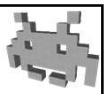


- In C#, a *property* has a syntax making it look like a field (you can read or assign it) but it's actually *getter* and *setter* methods
  - `obj.xx = 3` *...means...* `obj.set_xx( 3 )`
  - `foo = obj.xx` *...means...* `foo = obj.get_xx()`
- In Unity, a *component* is a generic something attached to a GameObject
  - `GameObject g;`
    `g.getComponent< type >()`
    returns component of required type (if it exists)

base class for everything in the game

43

Nodes of a scene-graph in Unreal Engine USceneComponent

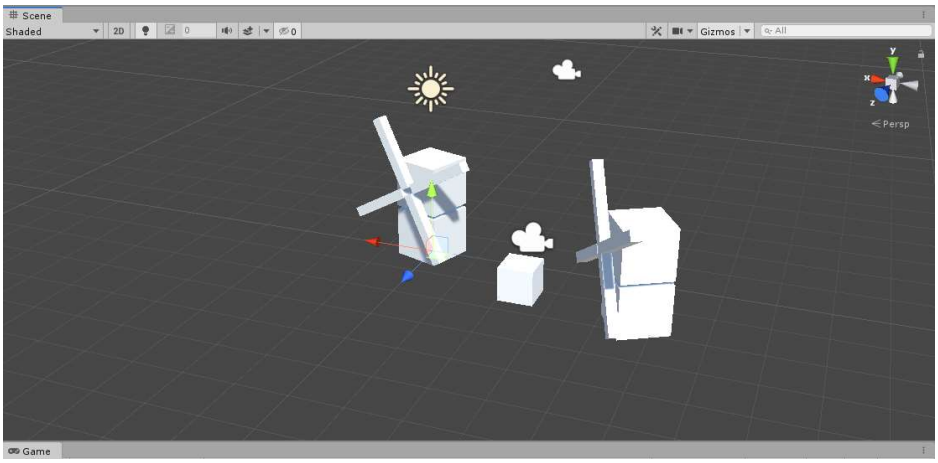


A node within a graph with:

- link to parent / children:
  - getParentComponents
  - getChildComponent(index)
- associated stuff to it: UPrimitiveComponent (subclass)
  - for models, physical bodies, etc
- Local Transform: (fields)
  - RelativeLocation , RelativeRotation, RelativeScale
- Global Transform: (methods)
  - GetComponentTransform() /* return transformation */

44

A demo with Unity



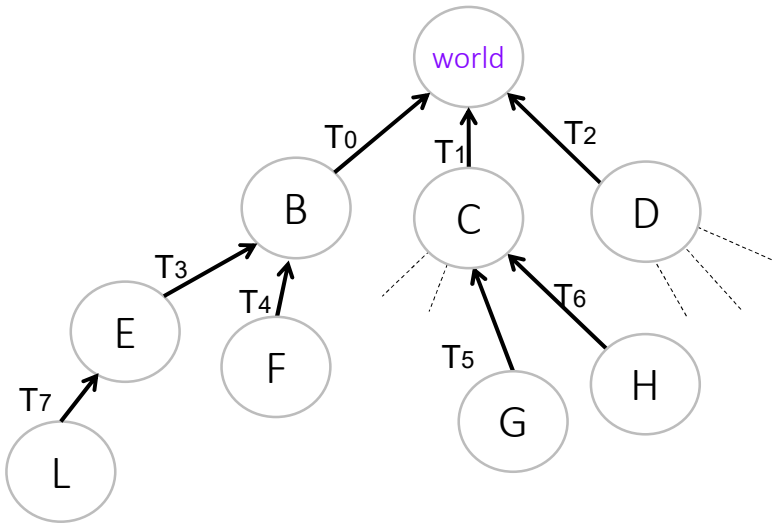
45

Mechanisms for common shared subtrees

- In Unity: see “Prefabs”
- In Unreal: see “BluePrints”

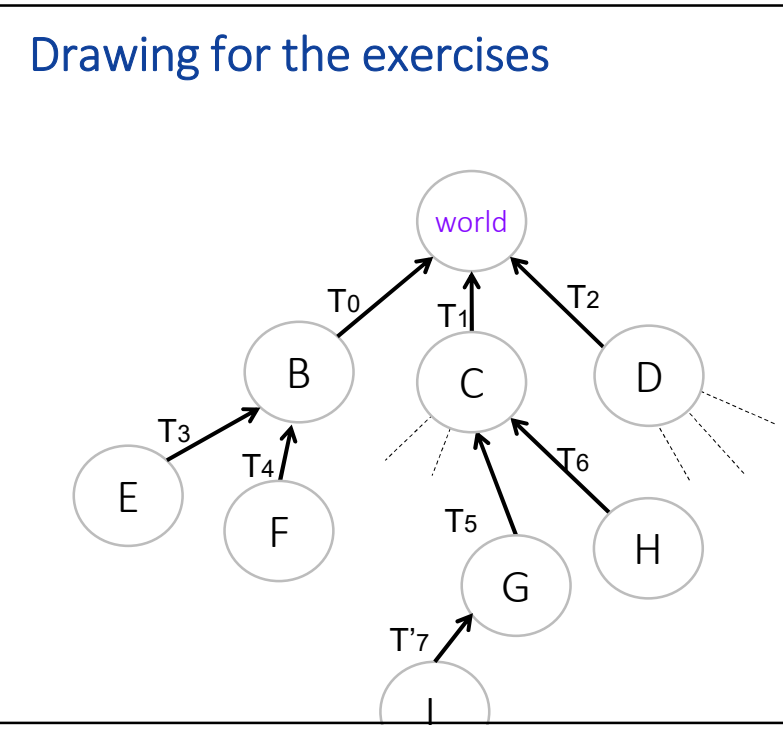
46

Drawing for the exercises



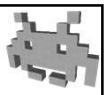
47

Drawing for the exercises



48

Exercises 1/2



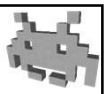
What is the new transform T'_7 which should subtitute T_7 if…

- …node **L** il reattached as a child of **D**, leaving its position in *world space* unaffected (e.g. by a scener, or a script)
- …node **D** is attached under node **L**, without affecting its world space position.
- …the object in node **L** must be moved 1 unit on the right in view space (camera is in node **C**)
- …the object in node **L** must be moved by 1 unit ON ITS RIGHT (i.e. that object in must be displaced by a new transform T applied in post-transform space)

Note: these kinds of problems are silently solved by Unity all the times (in the scripts & when user manipulates the the GUI)

49

Exercises 2/2



- Report the *global transform* of node **L**
- I place a camera in node **H**:
  report the View Transform for this scene
- What does it mean to apply a translation (0,2,0) to L …
  1. in L Space (the local space of L)?
  2. in World space?
  3. in View Space?
- Say T7 is the identity, and the camera is in H:
  how to modify T7 to get the case 1,2 or 3?
- Find the origin of space E in space H, and viceversa
- A microfone is in (the origin of) node E, and a speaker is in (the origin of) node H. Find the distance from the mic to the speaker

50