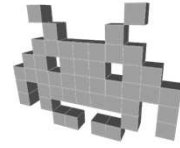


Master Videogames - Verona

the Scene Graph



Marco Tarini



1

Recap: 3D Spatial Trasforms



- Math functions
 - input: point / vector / versor
 - output: point / vector / versor
- Typically:
 - Scaling + rotation + translation
- Modelling:
 - scaling up or down / size
 - isotropic (uniform) o anisotropic (deforming)
 - orientation in space / rotation
 - movement / position (traslation)

} Thus, can be applied to e.g 3D models (apply it to every vertex position and normal...)

3

Recap: internal representations of 3D Spatial Trasforms



- Many possible ones, but typically:
 - translation (vec3) + scale (float or vec3)
+ rotation (3x3 matrix / quaternion / axis+angle / Euler angles)
- Regardless, we assume a transform is:
 - Light in memory (few tens of bytes)
 - Quick to apply (even to large, complex models)
 - It's done on-the-fly during rendering on the GPU
 - Quick to:
 - Interpolate with another (aka "blend" / "mix" / "lerp")
 - invert (find the opposite transform)
 - cumulate (find $A*B$) («they are closed w.r.t. = composition»)
- Always interpretable as a change of Reference frame
 - holds by def. for all the ones we care about: the "affine" ones

4

Recap: transformation associated to an object in the scene



- Any object associated to a spatial location in the game is given its transformation, which goes
- From:
 - **local space** *a.k.a.*
 - **object space** *a.k.a.*
 - **pre-transform** space
 - *a.k.a.* «castle» space /
«hero» space /
«camera» space /
«chainsaw» space /
«bazooka» space etc
- To:
 - **global space** *a.k.a.*
 - **world space** *a.k.a.*
 - **post-transform** space

5

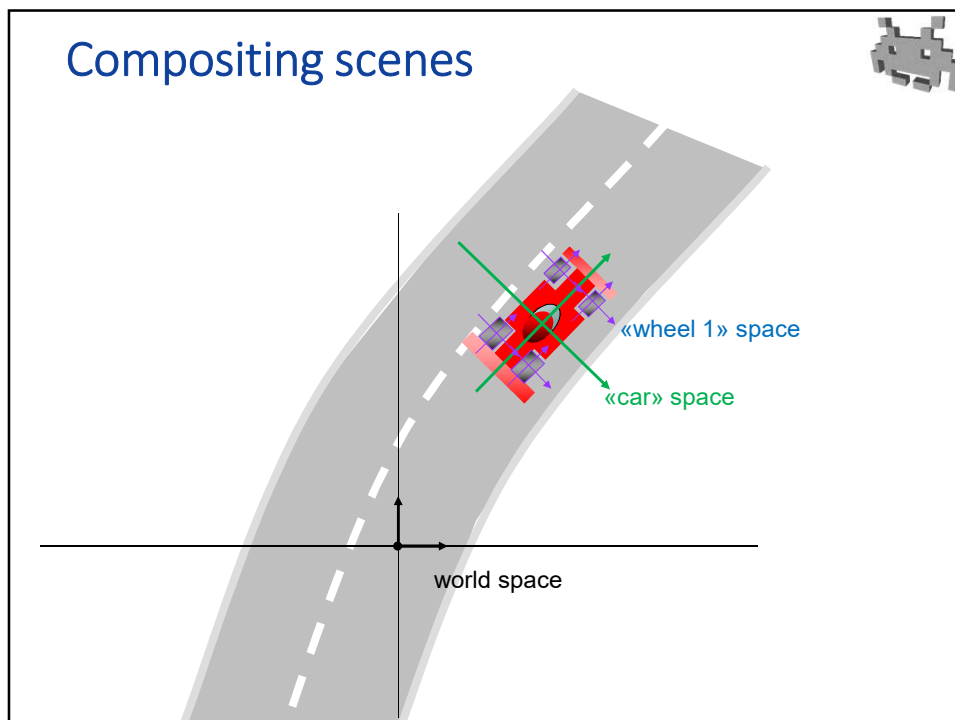
Composite scenes: hierarchical transformations



- So far, we assumed that the transform of each object goes from local to global in one step
- In reality, scene is constructed hierarchically
- Objects are made of sub-objects
 - a city is made of houses made of walls made of bricks
 - a «hat» sits on an «head» which sits on a «character» which sits in a «spaceship» moving across the «scene»
- Also: different instances of one object can appear in multiple locations of the scene

6

Compositing scenes



8

Scene graph

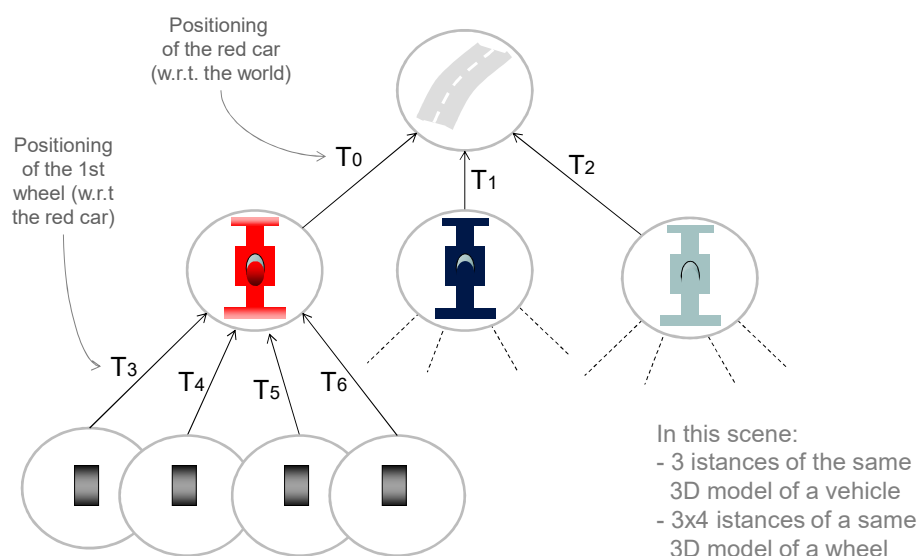


A tree (i.e. hierarchical structure)

- Each nodes: a space (a reference frame)
- To each node we associate:
 - Instances to stuff...
(3D models, lights, cameras, virtual microphones spawn points, explosions etc...)
- Root node: world space
- On the arches:
 - the local trasforms

9

Scene graph



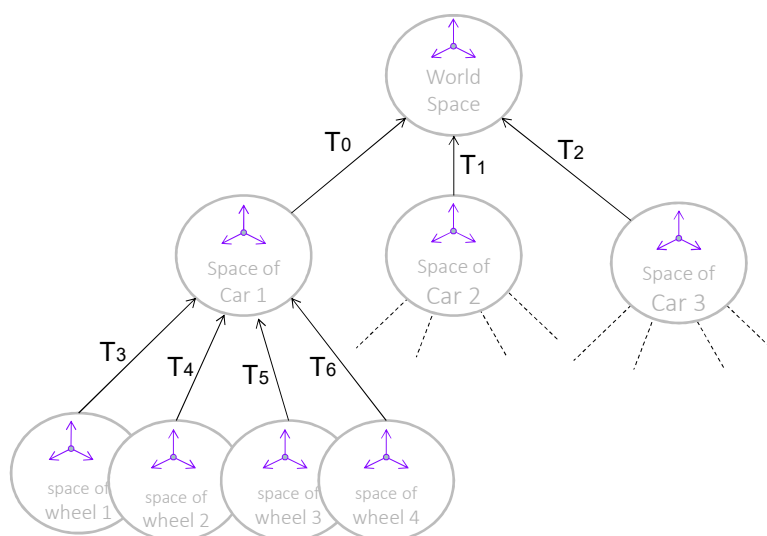
10

Local VS global Transform

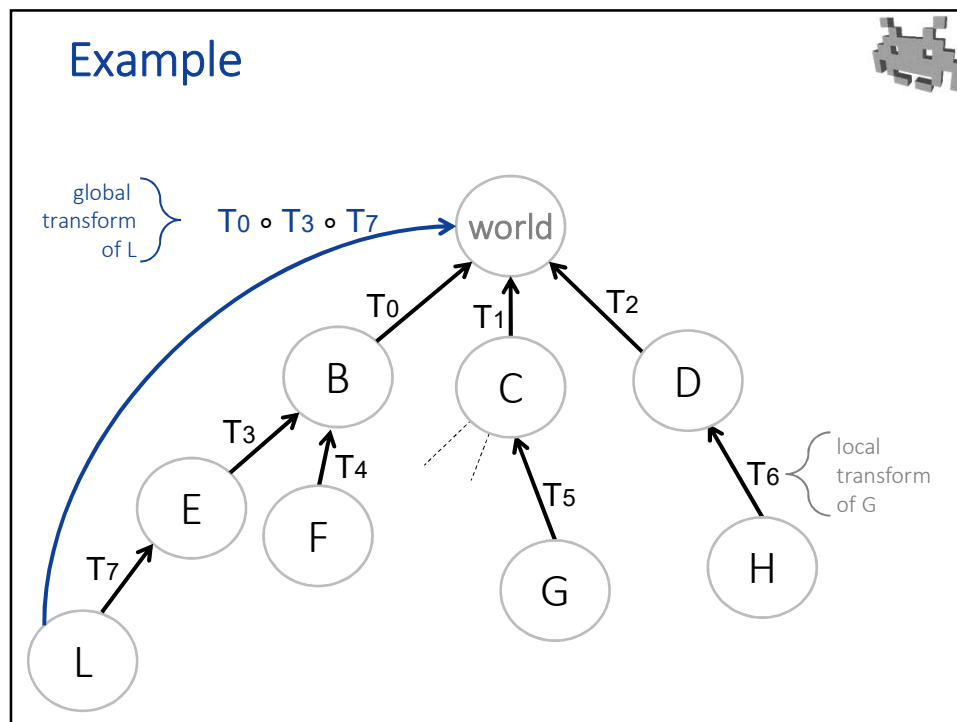
- Local transform (a.k.a. «relative» transform)
 - from a node space to its parent space
- Global transform (a.k.a. «absolute» transform)
 - from a node space to world space
 - obtained by: *cumulating* local transforms!
- benefit: changing the transform associate to a node affects its entire subtree!

11

Scene graph



12

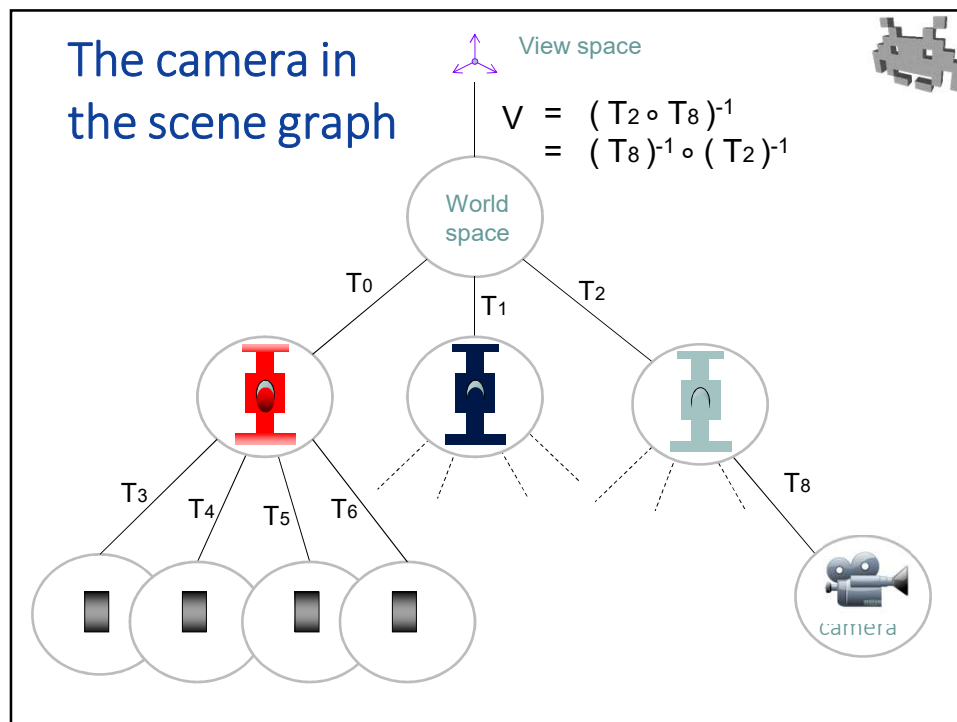


15

The camera in the scene graph

- Camera:
 - just another element sitting somewhere the scene-graph
 - for the scene to be rendered, there must be a camera somewhere in the graph!
- the «camera space» (object-space of the camera obj) is special
 - aka View Space (Screen Space is similar)
 - in CG, the View Transformation V
 - = inverse of Global Transform of camera node
 - = from World-space to View-space
 - V is used in the rendering to determine where objects end up on the screen.
- Camera animations = move camera = change V
 - e.g. by script

16



17

Authoring a 3D scene

1/3

- Eg, as a sub-task of the Level Design
- The two task are different:
 - 3D modellers make «scene props»
 - the **3D models** to be combined
 - (+ **textures** etc)
 - «sceners» compose the scene
 - they assemble the props into a **Scene Graph** (completed with a local transform)

[DEMO]

18

Authoring a 3D scene 2/3



- Additional assets:
 - a **Collision Mesh** (a **Geometry Proxy**)
 - for each scene-prop
 - can this be made automatic? Possible, not easy
 - needed for: physics, visibility computation, AI, and all sorts of gameplay reasons...
 - a **Navigation Mesh** (aka **AI mesh**)
 - 1 per the entire scene
 - needed by: AI (routing)
 - can this be made automatic? Possible, not trivial

19

Authoring a 3D scene 3/3



- Additional assets:
 - **Scripts** ?
 - task of the level designer
 - **Sky box**
 - **Outer terrain** mesh...
 - Ambient **sounds**
 - etc (data such as spawn points, etc)

20

Scene Graph as a data structure



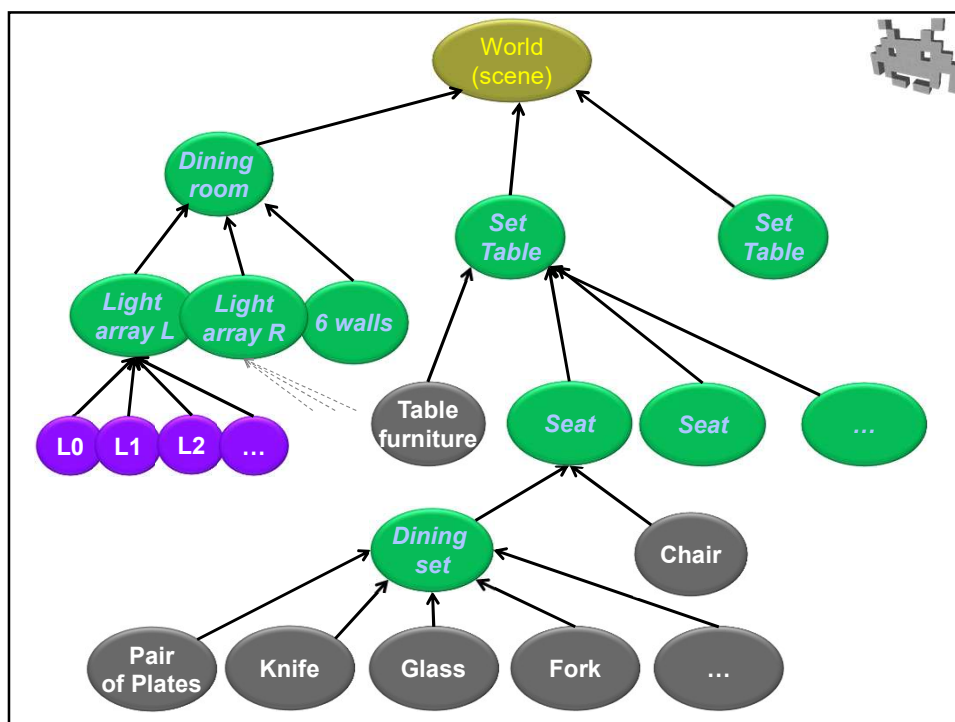
- Each engine / library adopts its own solution
- No standards
 - but file formats exists which can include a scene graph: e.g. COLLADA
- Typical concepts:
 - the Node class keeps the (local) transform to the father node (and / or childs, sibilings...)
 - also, links (eg. pointers) to instances / assets
 - global transforms are computed on demand
 - camera node is a special node
 - some mechanism is needed for repeated sub-trees

21

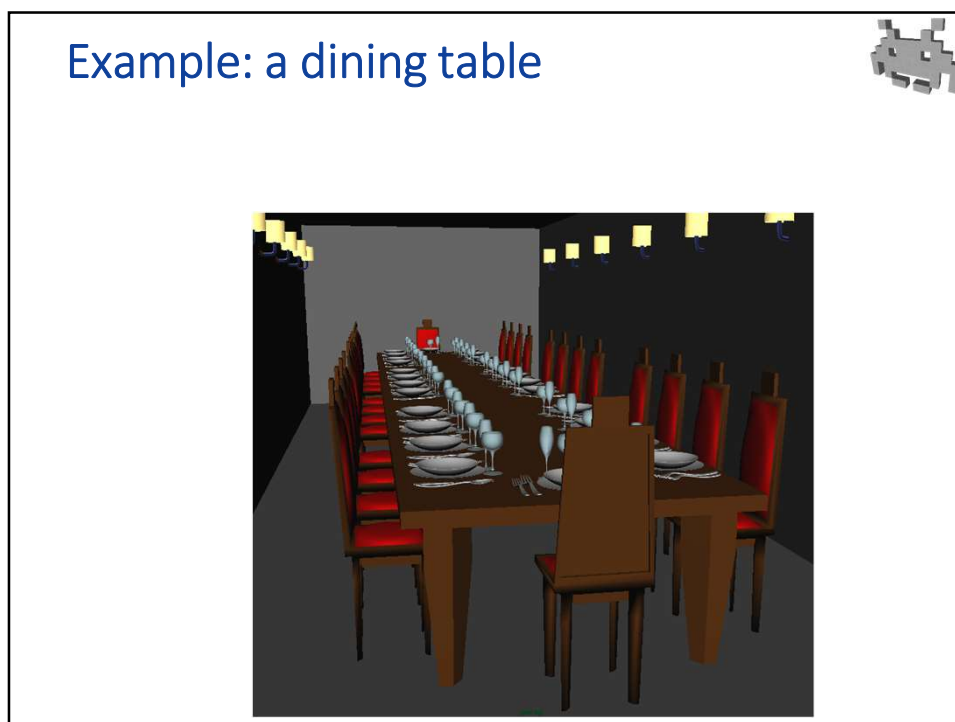
Example: a dining table



23



24



25

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!

26

Nodes of a scene-graph in unity GameObjects & Transforms



- The Transformation actually stores the local transf:
 - **localPosition, localRotation, localScale**
 - remember: goes from this node to father node
- the Global transformation can be accessed via the properties:  feels like assigning / reading a field, actually means invoking setters/getters (C# trick)
 - **position, rotation, scale** (“**global**” is left implicit)
 - what does getting / setting them really do? exercise!
 - it doesn’t work for “scale”! why? (because anisotropy)

28

Digression on unity : properties and components



- Properties (C# mechanism)
it feels like a field (you can read or assign it)
but it's actually a getter and setter method
 - `obj.xx = 3` ...means... `obj.set_xx(3)`
 - `foo = obj.xx` ...means... `foo = obj.get_xx()`
- Components (Unity library mechanism)
 - A generic something attached to a GameObject
 - `GameObject g;`
`g.GetComponent< type >()`
returns component of required type
(if it exists)

base class
for everything

29

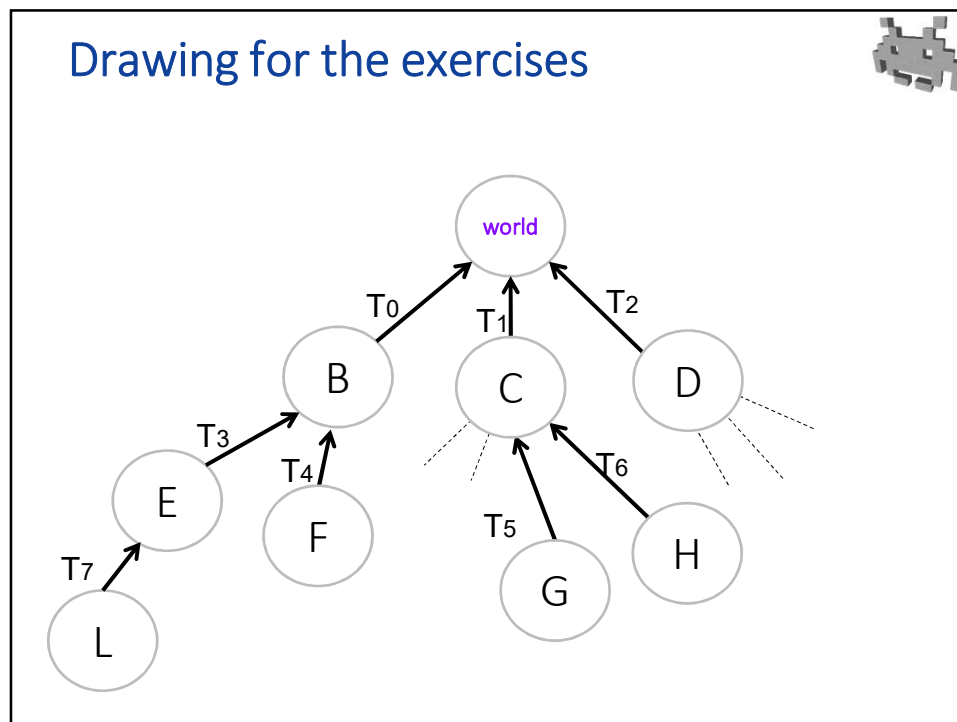
Nodes of a scene-graph in Unreal USceneComponent



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 */

30



31

Exercises 1/2

- Give the *global transform* of node L
- I place a camera in node H:
report the View Transform for the scene
- What does it mean to apply a translation (1,0,0) 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 1,2,3 cases?
- Find the origin of space E in space H, and viceversa
- A microphone 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

32

Exercises 2/2



What is the new transform T_7' which should substitute T_7 if...

- ...node **L** is 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 (assume T_7 is currently the identity)
- ...the object in node **L** must be displaced by a new transform **T** applied in post-transform space.

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

33

Mechanism for shared subtrees



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

34