ESCOLA DE NOVES TECNOLOGIES INTERACTIVES

centre adscrit a:

UNIVERSITAT DE BARCELONA

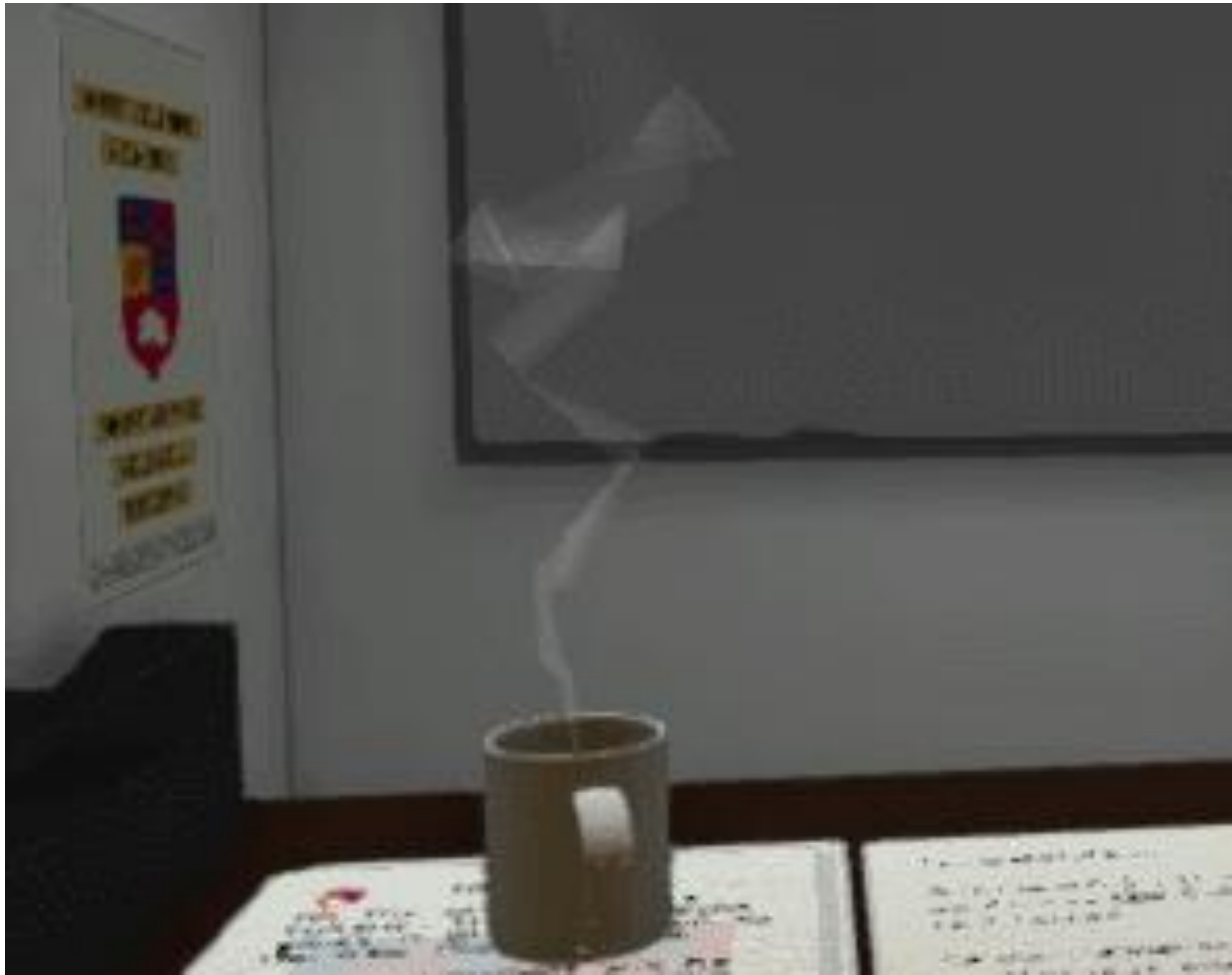
Animation Foundations

05. Introduction to procedural animations

+ Exercises on rotations

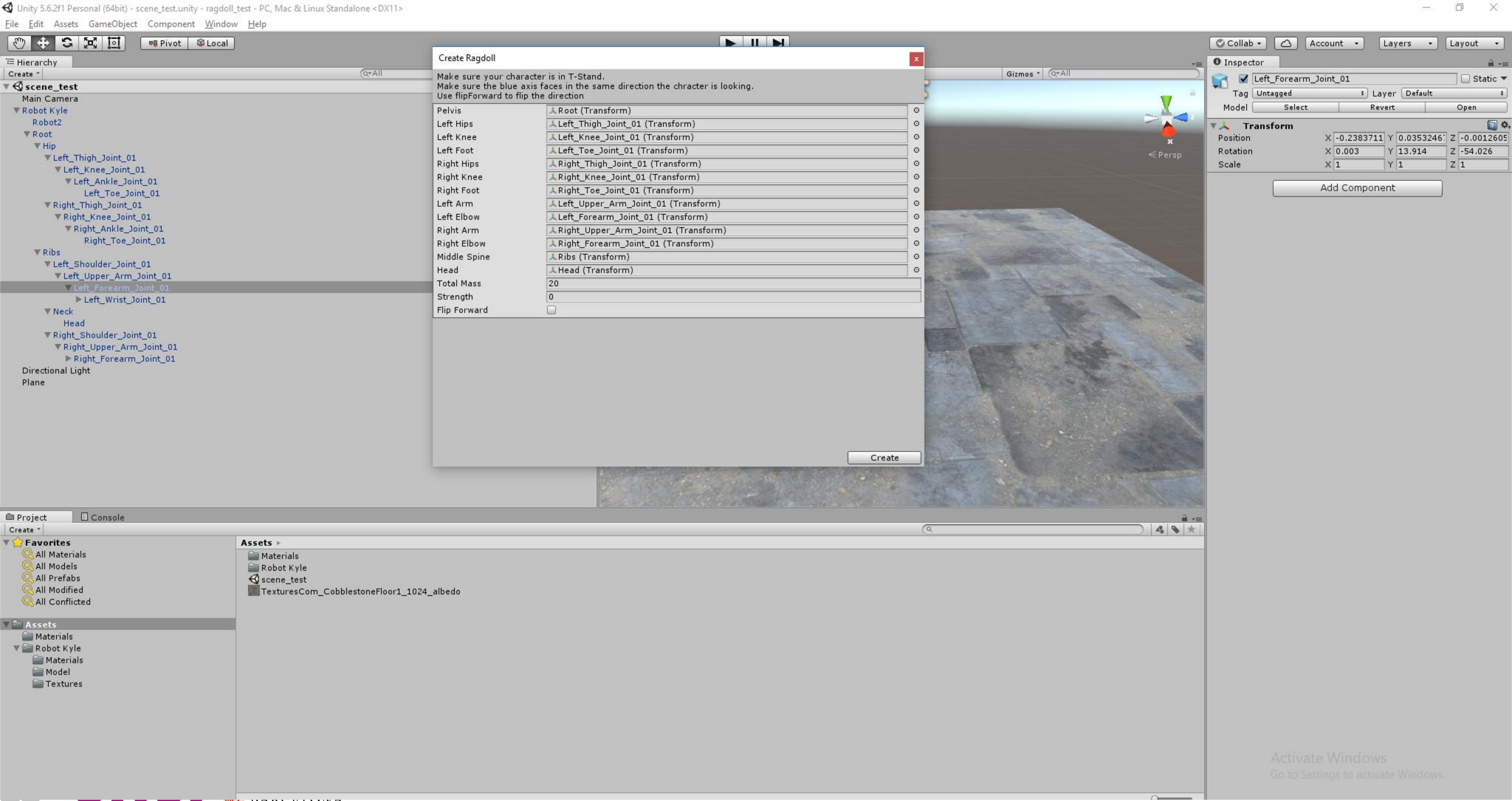
Kinds of animations

- Physically-based Animations
  - Tissue
  - Water
  - Smoke
  - …
- Ragdoll physics
  - Ragdolls
  - Physics-based character animation
- IK



Examples

- Ragdoll example in unity
- Videogames
    - A Bud’s life
    - Gang Beasts



Create Ragdoll

Make sure your character is in T-Stand.
Make sure the blue axis faces in the same direction the chracter is looking.
Use flipForward to flip the direction

Pelvis	Root (Transform)
Left Hips	Left_Thigh_Joint_01 (Transform)
Left Knee	Left_Knee_Joint_01 (Transform)
Left Foot	Left_Toe_Joint_01 (Transform)
Right Hips	Right_Thigh_Joint_01 (Transform)
Right Knee	Right_Knee_Joint_01 (Transform)
Right Foot	Right_Toe_Joint_01 (Transform)
Left Arm	Left_Upper_Arm_Joint_01 (Transform)
Left Elbow	Left_Forearm_Joint_01 (Transform)
Right Arm	Right_Upper_Arm_Joint_01 (Transform)
Right Elbow	Right_Forearm_Joint_01 (Transform)
Middle Spine	Ribs (Transform)
Head	Head (Transform)
Total Mass	20
Strength	0
Flip Forward	□

Create

Left_Forearm_Joint_01
Tag Untagged Layer Default
Model Select Revert Open

Transform

	X	Y	Z
Position	-0.2383711	0.0353246	-0.0012605
Rotation	0.003	13.914	-54.026
Scale	1	1	1

Summary of previous courses on rotations:

- Rotations in 2D
  - Angle
  - Matrix
- Rotations in 3D
  - Euler Angles
  - Yaw-Pitch-Roll
  - Axis Angle
  - 3x3 Matrix

Today, we use the stuff that has imagination:

- Reminder rotations in 2D
  - Angle
  - Matrix
  - Complex Numbers
- Introduce New method for Rotations in 3D
  - Euler Angles
  - Yaw-Pitch-Roll
  - Axis Angle
  - 3x3 Matrix
  - Quaternions

Rotations

- With complex numbers
- With quaternions

We want to have:

- Compact representation
- Simple calculation
- Robust composition
- Robust interpolation

Exercise 1

Find the offset angles between target1 and tracker.

Then make target1 align with tracker.

- Make it with object “tracker” and target1 “rectangle1”
- Use angle axis to find explicitly the angle offsets.

Exercise 2

Make target1 align with tracker.

- Use one ligne of code (use the quaternion that corresponds to the offset rotation)

Then, make it align with tracker, but slowly in time.

- Use method **Quaternion.AngleAxis**
- Use method **Transform.Rotate**

Exercise 3

Make target1 follow tracker while keeping the offset.

1. Make it with object “tracker” and target1 “rectangle1”

Use exercise 2 and apply a quaternion transf. to it

2. Imagine “tracker” is an HMD tracker, and apply it also to the robot’s head

3. Apply it to the robot’s head and to the virtual camera

Exercise 4

Make target2 follow the transformations of target1, but in such a way that it is aligned with the tracker

How can you find the right offset?

Exercise 5

Write your own Quaternion class that:

- Always keeps values normal
- Can multiply quaternions
- Can invert quaternions
- Can convert from axis angle
- Can convert to axis angle
- Optionally, gives a warning if it is rotating more than 180º

- Check that exercise 4 still Works when using it

To design the class, imagine that in the future you might want to encapsulate it in a .dll

- Base it solely on the Mathf library
- Make it independent from gameObject