Department of Computer Science

Augmented Reality 2018

06-02-2017

Group assignment 1

Deadline: **13-02-2017**

Preparation

- Download and print the templates for image targets *spaceship_set* and *earth* from Blackboard (ar_assignment1_templates.zip).
- For all tasks, it is **mandatory** to set the *World Center Mode* of your `ARCamera` to *CAMERA*.

Transformations and coordinate systems

Tasks

a) **1pt** Generate individual image targets from each of the images of the template. Use exact measurements and be consistent with your units.

> Show your image files and explain how you measured the size of the printouts and how these measurements relate to the image files.

b) **2pt** You will realize that some of the image targets have a very bad *Augmentable* rating in the Vuforia target manager. Add visual features to your image targets or completely rework them. You can follow this guide. Keep it close to what the targets depict, e.g., the spaceship should still be recognizable as such when looking at it. Every target has to have a rating of at least 3 stars in the end.

> Why was it initially hard to track for Vuforia and why did your features improve it? Why wasn't it necessary for some of the targets?

0.5pt Achieve an *Augmentable* rating of at least 4 stars per target while keeping them close to what they depict.

c) **2pt** Create a scene with the space shuttle and the landing lane. Attach a small patch (e.g., Quad) to the space shuttle and assign it a new material, using the *Unlit* → *Color* shader. Set the

color of the material at runtime depending on how much the space shuttle is aligned with the landing lane. Use the dot products of the base vectors of both GameObjects. The patch color should go from red (not aligned) to green (completely aligned).

d) **2pt** Create a scene with the space shuttle and the large earth image target. Attach a GameObject to the nose of the space shuttle. Transform the nose's `localPosition` to get from the space shuttle's local coordinate system to the earth's local coordinate system. Do this by first creating a `Matrix4x4` (based on the GameObjects' model matrices) that transforms between the shuttle's and earth's local coordinate systems.

> Imagine having to transform very many points. Which of the steps above have to be done once per frame and which ones for each point?

1pt Check whether or not the nose is over the (round) image of the earth, using the local xyz-coordinates you just computed. You can achieve that by implementing cylindrical boundaries for the local coordinate system, i.e., testing how far away (x, z) is from the origin and checking whether y is below a threshold. You can choose an appropriate cylinder height, i.e., maximum y value. If they are within the boundaries, display "North" or "South" depending on whether the nose is over the northern or southern hemisphere.

e) **2pt** Create a scene with the Millennium Falcon and an enemy ship (TIE Fighter or Interceptor). Assign a button or key stroke to shoot a laser from the Falcon to its viewing direction. Check whether the ray hits the enemy ship and indicate a hit (e.g., with an explosion). Visualize the ray: Generate a `OnRenderObject` method and use `GL.Begin(GL.Lines)`.
0.5pt Shoot two lasers from the two front cannons (see Figure 1). Instead of using child GameObjects for your cannons, define the local positions of the cannons and use the Falcon's position and rotation to get the cannons' world positions.
1pt Shoot a laser from the top cannon (see Figure 1). Do not use child objects, but define your local coordinates in code. The top cannon should be 2.5cm above the spaceship. Define yaw and pitch variables to adjust the angle of the top cannon and apply the rotation to the direction of the laser. You can use `Quaternion.Euler` to create a rotation from your angles.

Min / Total = **6.0pt / 12.0pt**

TIPS

- Even though it is not required in this assignment, we recommend to always attach some 3D objects to your image targets to quickly see whether they are tracked correctly by Vuforia. The more accurate your measurements, the more accurate the spatial relations will be, which is very essential for this assignment. When being in play mode, switch to the scene view of Unity to verify that the image targets have the correct spatial relations while being tracked. We recommend to have the scene view always visible also during play mode.
- You can use simple image editing software to add visual features to your image targets or paste small images into it. It does not have to be pretty. A simple way is to incorporate areas from the (very optimized) Vuforia object scanning target.
- Unity has a ray casting functionality that you can use for your lasers: Attach a *box collider* to the enemy ship (make the bounding box a bit bigger to compensate for inaccuracies). Use `Physics.RayCast` to cast a ray and check if it hits the bounding box.
- It is not mandatory to use matrices for the Millennium Falcon task. You can apply the Falcon's `transform.rotation` to the local positions of the cannons and shift it by the Falcon's `transform.position`. This effectively ignores the scale of the Falcon.
- You can use this as an orientation for the Millennium Falcon task:

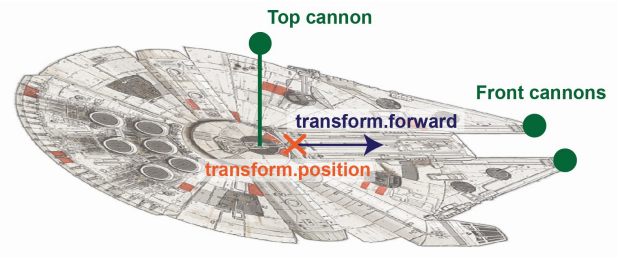

Figure 1: Local positions of cannons on the Millennium Falcon.