samhoud|media

Short document

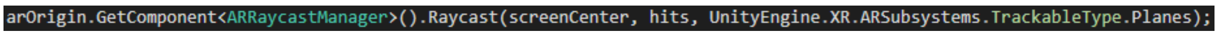
Building an AR racing car!

This document gives a short explanation of the design choices that were made for the AR racing car assignment.

Checklist:

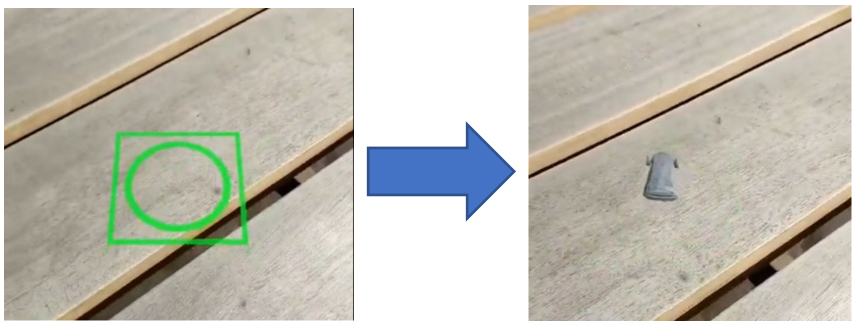
As a user I want to be able to:

- See the horizontal planes in the AR world
    - Through raycasting we check whether the center of the screen is pointing to a flat surface. A code snippet of this can be found below.



```
arOrigin.GetComponent<ARRaycastManager>().Raycast(screenCenter, hits, UnityEngine.XR.ARSubsystems.TrackableType.Planes);
```

- Place the car object on a detected plane
    - The placement indicator is only set to active when a plane is detected. If currently there is no plane detected, the placement indicator is simply disabled. A user can place the car by tapping on the screen while the placement indicator is active.

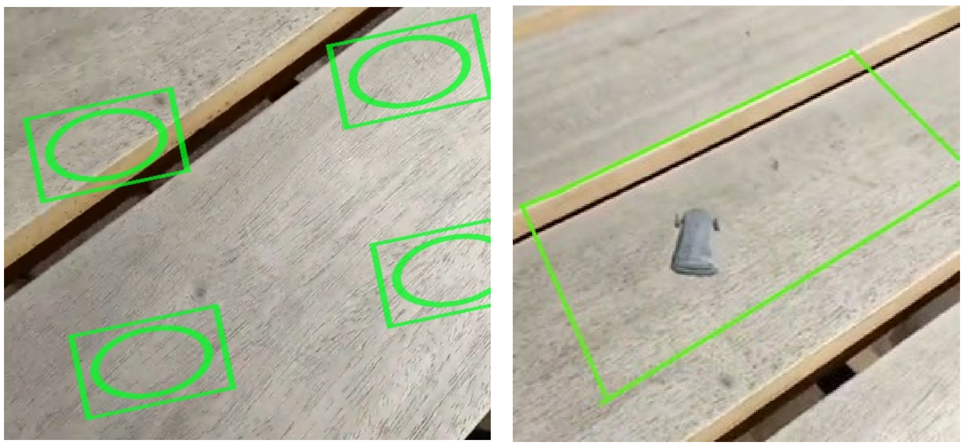


- Control the car in the AR world with onscreen joysticks
    - User can do this through the Mobile Joystick that can be found in the Unity Standard Assets package. This controller uses CrossPlatformInput, simplifying input handling for cross platform projects. The chosen car is part of the same asset's package and includes a user control script using CrossPlatformInput. The car has been set to a toy-like size to give off the feeling it's a desk toy.
- Use the app on an Android or iOS device
    - An .apk file has been added to the Github repository, so a user can run the project on an Android device.

Future improvements:

While working on this project, I immediately thought of possible future improvements for this little project. Even though I did not implement them (since it was not included in the user story), I decided to list them here:

- Multiple cars, coloring, shading, scaling and reflection options
- Making a game using the car controlled in the AR world (like the Nokia classic Snake or just collecting coins)
- Defining playground a bit better: currently once the car is placed in the AR world, its vertical point in space does not change. This means the car can drive off the real-life horizontal plane.
    - Solution 1: Let users define a horizontal plane and use that as the interaction area. The disadvantage of this method is that it requires more steps of the users. The image below illustrates the approach. On the left, the camera detects four horizontal planes. On the right, the car gets spawned and can interact in the AR world within the set borders.



    - Solution 2: Check immediate surroundings of car every frame. Although this might be computationally more expensive, raycasting in a lot of different directions can give the software a better approximation of the 3D space. By doing this, the car is not necessarily confined to a space like the previous example, while still being limited by walls or height differences. The disadvantage of this approach is that users need to look at the car continuously in order to raycast its immediate surroundings.