

Minor - algemeen

Opdrachten periode 4

Leerjaar:	1
Crebonummer:	25604
Sector:	Software developer

INTRODUCTIE

In de vorige periode heb je kennis gemaakt met de minors die in de opleiding software development worden aangeboden, namelijk:

- Verdieping programmeren
- Digital Signage
- IOT
- XR

Inmiddels heb je ook al een keuze gemaakt (of doe je binnenkort) in welke minor je je het meest wilt verdiepen. In minor-algemeen gaan we echter wel verder met alle minors, zodat je van deze minors in ieder geval de basiskennis en vaardigheden opdoet.

In deze periode gaan we in groepen aan de slag en maken we een totaalproduct dat bestaat uit een combinatie van een aantal deelproducten. Elk deelproduct is een opdracht per minor. De afzonderlijke opdrachten staan in aparte hoofdstukken uitgelegd. Het geheel wordt aan het eind van de periode aan de groep gepresenteerd.

TIJDSPLANNING

Week 1		
Minor algemeen, week 1	0,5 uur	<u>1</u> Onderwerp en Start
	1,5 uur	Zelf per groep in te plannen
Week 2		
Minor algemeen, week 2	4 uur	Zelf per groep in te plannen
Week 3		
Minor algemeen, week 3	4 uur	Zelf per groep in te plannen
Week 4		
Minor algemeen, week 4	4 uur	Zelf per groep in te plannen
Week 5		
Minor algemeen, week 5	4 uur	Zelf per groep in te plannen
Week 6		
Minor algemeen, week 6	4 uur	Zelf per groep in te plannen
Week 7		
Minor algemeen, week 6	4 uur	Presenteren van het geheel

LES 1 - 2 UUR – ONDERWERP EN START

INTRODUCTIE

Voor de vier minors is er een opdracht in dit lesmateriaal opgenomen. De vier opdrachten hebben allemaal met hetzelfde onderwerp te maken en vormen zo een totaalpakket. Je maakt gedurende de periode waarin de lessen gepland staan met jouw groepje de vier opdrachten en zorgt ervoor dat er een samenhang is. Deze samenhang zit bijvoorbeeld in het uiterlijk van de verschillende onderdelen en je probeert de applicaties tot één geheel te maken.

Het onderwerp waar alle opdrachten mee te maken hebben is een fastfood Restaurant, zoals MacDonald of Burgerking. Hieronder staat globaal aangegeven wat je per opdracht gaat maken:

- | | |
|----------------------------|--------------------------------------|
| • Verdieping programmeren: | webapplicatie voor een kassa-systeem |
| • Digital Signage: | bestelzuil |
| • IOT: | klanten-oproepsysteem |
| • XR: | AR hamburger op tafel maken |

De verdere beschrijving en uitleg van deze opdrachten vind je in de volgende hoofdstukken.

WERKVORM

Deze opdrachten ga je in een groep van vier maken. De samenstelling van de groep is niet zo belangrijk, maar het kan zinvol zijn om een groep samen te stellen waarvan de groepsleden verschillende minors gekozen hebben als verdieping. Er is dan voor elke opdracht een specialist in het team.

Na de introductie van de docent en het samenstellen van de teams ga je eerst een planning maken voor de gehele periode. Geef hierin aan wat de groep per week gaat doen en wie er aan welke afzonderlijke opdracht gaat werken. Laat de planning zien aan de docent als je deze klaar hebt. Hou er in deze planning rekening mee dat de laatste les voor presentaties bedoeld is. Het geheel moet dan dus klaar zijn.

In de laatste les gaat de groep het totaalresultaat presenteren aan de gehele klas. Hierbij vertelt en demonstreert elk teamlid een deel van het totale product (iedereen komt dus aan het woord).

OPDRACHT 1 – HET KASSASYSTEEM (VERDIEPING PROGRAMMEREN)

INTRODUCTIE

Voor het fastfood restaurant moet er een kassasysteem gebouwd gaan worden. Niet alle klanten van het restaurant zullen gebruik willen maken van een digitale bestelzuil, sommige mensen willen gewoon kunnen bestellen en afrekenen aan de kassa. Dit kassasysteem

wordt gebruikt door een medewerker van het restaurant om bestellingen op te nemen en af te rekenen.

Het fastfood restaurant heeft al een basis voor het systeem laten bouwen door een externe developer maar het is aan jou om het systeem helemaal werkend te krijgen.

Het kassasysteem moet online in een webbrowser kunnen werken dus het is een webapplicatie waarbij we gebruik gaan maken van een decentrale SQLite database. Deze database gaan we uitlezen met behulp van PHP en we gaan een combinatie van HTML, Javascript en PHP gebruiken om het kassasysteem op te bouwen. Je zult zien dat de mix van deze 3 talen/technieken heel krachtig kan zijn.

Item	Price	How many	Price
Hamburger 1.75	3.5	2	3.5
Milkshake 2.4	2.4	1	2.4
Frenchfries 2.25	4.5	2	4.5
Cheeseburger 1.5	3	2	3
Chickenburger 3.25	3.25	1	3.25
Fishburger 2	2	1	2
Total amount for this order: 18.65		Check out	

THEORIE

Zoals je hierboven hebt gelezen gaan we een kassasysteem bouwen voor het restaurant. Dit systeem moet door iedere medewerker gebruikt kunnen worden dus het moet een makkelijk te bedienen webapplicatie worden.

Het idee is dat er per product (op dit moment zijn dat er 6) een button is waarop geklikt kan worden. Door b.v. op de button Hamburger te klikken voegt de medewerker 1 Hamburger aan de bestelling toe, het systeem rekent automatisch de totaalprijs voor alle Hamburgers uit en berekend ook direct de totaalprijs voor de hele bestelling. De velden 'How many', 'Price' en 'Total amount' kunnen NIET met de hand ingevuld of aangepast worden. Vervolgens moet de medewerker op de knop 'Check out' klikken om de bestelling af te rekenen.

Op dit moment is het nog niet nodig dat de bestellingen opgeslagen worden in het systeem. We gebruiken dus alleen een database om gegevens uit op te halen.

Om de applicatie goed te laten werken is het belangrijk dat je eerst jouw werkomgeving klaarzet.

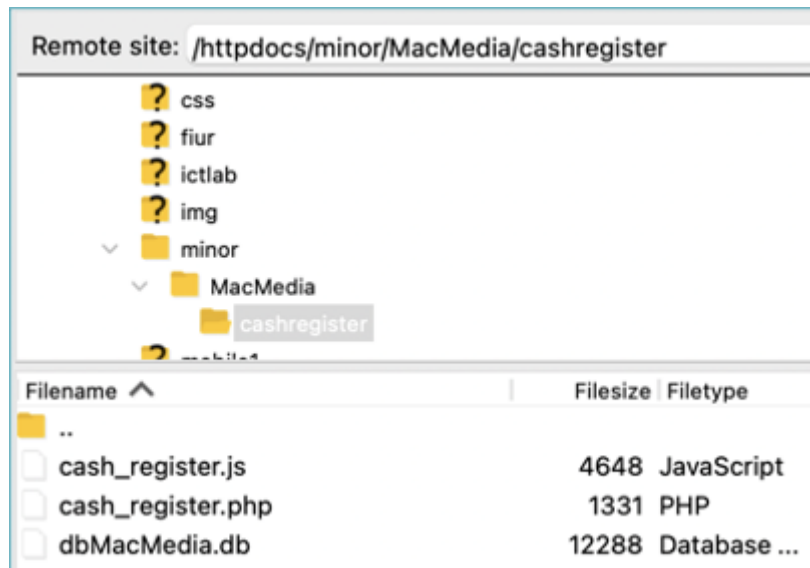
OPDRACHT 1.1 WERKOMGEVING KLAARZETTEN

Voer het volgende uit:

1. Maak op de webserver een map aan met de naam 'cashregister'.
2. Download het bestand 'cashregister.zip' van Digitale Lesstof.

3. Pak het zip-bestand uit kopieer de inhoud in de zojuist gemaakte map op de server.

Als je dit allemaal hebt uitgevoerd dan zou jouw werkomgeving er uit moeten zien zoals op onderstaande afbeelding.



THEORIE DATABASE KOPPELEN

De database die je gaat gebruiken heet 'dbMacMedia.db'. De database bevat 1 tabel genaamd 'snacks' en de structuur van deze tabel is eenvoudig, hieronder bekijken we deze structuur kort.

Table: snacks

	ID	snackName	snackPrice
	F...	Filter	Filter
1	1	Hamburger	1.75
2	2	Milkshake	2.4
3	3	Frenchfries	2.25
4	4	Cheeseburger	1.5
5	5	Chickenburger	3.25
6	6	Fishburger	2.0

Het id is de unieke aanduiding van een rij. Dit is altijd noodzakelijk in een database, iedere tabel heeft altijd een ID-kolom. De tabel is zo ingesteld dat het id automatisch door de database wordt ingevuld.

In de kolom 'snackName' staat de naam van de snack en in de kolom 'snackPrice' staat de prijs.

Om de database te kunnen gebruiken moet je in de PHP-pagina een verbinding maken met de database. Dit doe je op dezelfde manier als dat je in de Unity module ook hebt gedaan.

OPDRACHT 1.2 DATABASE KOPPELEN

Open het bestand 'cash_register.php'. In dit bestand staat al behoorlijk wat code maar deze code werkt nog niet. Aan jou de opdracht om te zorgen dat dit een werkend kassasysteem gaat worden.

1. Op regel 12 maken we de verbinding met de database. Zoek uit wat de naam van de database is die je moet gebruiken en pas 'xxxxxx.xx' aan naar de echte naam van het database-bestand.
2. Op regel 16 halen we alle gegevens op die in de tabel staan. Dit doen we met een 'SELECT' statement waarbij we met '*' aangeven dat we alle gegevens willen ophalen. Zoek uit wat de naam van de tabel is die je moet gebruiken en pas 'xxxxxx' aan naar de echte naam van de tabel.

THEORIE GEGEVENS UITLEZEN/KASSASYSTEEM OPBOUWEN

Op regel 22 van 'cash_register.php' gaan we beginnen met het uitlezen en verwerken van de gegevens uit de tabel. Alle gegevens, ook wel resultaten genoemd, worden opgeslagen in de variabele \$result. Aan ons de opdracht om de gegevens uit \$result te halen zodat we ze kunnen gebruiken in ons script. Dit gaan we doen met het 'while' commando.

```
while($snack = $result->fetchArray(SQLITE3_ASSOC)) {?>
```

In het commando zeggen we eigenlijk: Zolang er nog rijen met gegevens uit de variabele \$result komen, plaats deze in de variabele \$snack. \$snack zal dus steeds gevuld worden met de gegevens van de volgende snack die in de tabel staat.

Als eerste zal \$snack dus gevuld worden met '1, Hamburger, 1.75', vervolgens met '2, Milkshake, 2.4' en zo verder totdat er geen nieuwe rijen met gegevens meer uit \$result komen.

Door gebruik te maken van PHP kunnen we de gegevens uit de tabel gebruiken. De PHP-code '\$snack['snackName']' zal steeds gevuld worden met de naam van de snack, op dezelfde manier kunnen we '\$snack['snackPrice']' gebruiken om achter de prijs van de snack te komen.

Nu we dit weten kunnen we handig gebruik maken van deze techniek om ons kassasysteem op te bouwen. Binnen de while-lus, vanaf regel 23, gaan we voor iedere snack een eigen regel (<tr>) maken in het kassasysteem. In deze regel komt de button, het 'How many' veld en het 'Price' veld.

OPDRACHT 1.3 GEGEVENS UITLEZEN/KASSASYSTEEM OPBOUWEN

1. Op regel 24/25 maken we dynamisch een button aan. Deze button krijgt een onclick-event dat ervoor zorgt dat de Javascript functie 'addSnack' wordt uitgevoerd als er op de button wordt geklikt. We geven 2 parameters/argumenten mee bij de aanroep van

deze functie. De 2 argumenten halen we met PHP uit `$snack` en vervolgens plaatsen we de waarde hiervan in de HTML-code door het PHP 'echo' commando te gebruiken. De naamgeving van de 2 argumenten klopt echter nog niet, aan jou de opdracht om uit te zoeken wat de juiste naam van `$snack['snackXxx']` en `$snack['snackZzzzz']` is en om dit aan te passen in jouw code.

2. Doe dit ook voor regel 26 en 27 zodat de id's voor deze velden goed komen te staan.

THEORIE HET KASSASYSTEEM LATEN WERKEN

Tot nu toe hebben we gebruik gemaakt van HTML voor de structuur van de pagina en hebben we PHP gebruikt om gegevens uit de database op te halen en ze te gebruiken om de kassapagina dynamisch te genereren. Nu gaan we ervoor zorgen dat er ook echt iets gebeurt als er op de knoppen geklikt wordt. Hiervoor gebruiken we Javascript zodat de berekeningen in de browser (front-end) gedaan worden.

Het bestand 'cash_register.js' bestaat uit 3 secties.

- In sectie 1 worden de variabelen aangemaakt die we nodig hebben.
- In sectie 2 staat de hoofdfunctie van het script genaamd `addSnack`. Deze functie zorgt ervoor dat bij een klik op een button het aantal van een bestelde snack wordt opgehoogd, dat de prijs aangepast wordt en dat de totaalprijs van de hele bestelling aangepast wordt.
- In sectie 3 staat een functie die we gebruiken om de prijzen af te ronden. Door slim gebruik te maken van comments tussen onze Javascript code kunnen we de verschillende delen van het script uit elkaar houden, gebruik dit ook in jouw eigen code!

Laten we eens naar Sectie 2 kijken, de functie `addSnack`. We zien dat deze functie 2 parameters/argumenten verwacht namelijk 'snackName' en 'snackPrice'. In de functie maken we gebruik van een 'switch/case' constructie om te bepalen voor welke snack er iets gewijzigd moet worden. Als er op de button voor een Hamburger wordt geklikt dan moet er natuurlijk alleen maar iets gewijzigd worden voor de Hamburger en niet voor de andere snacks.

Op het moment dat de medewerker van het restaurant op de knop van Hamburger klikt moeten er 3 dingen gewijzigd worden namelijk:

- De waarde van het veld 'How many' moet opgehoogd worden met 1. In regel 9 lezen we de huidige waarde van het veld uit. Deze waarde zetten we in een variabele, vervolgens hogen we die waarde op met 1 (++) en plaatsen we de nieuwe waarde weer terug in het juiste HTML-veld.
- De waarde van het veld 'Price' moet opgehoogd worden met 1x de prijs van de Hamburger. In regel 14 bepalen we de totaalprijs door het aantal hamburgers te vermenigvuldigen met de prijs van 1 hamburger (deze prijs is als `snackPrice` meegegeven bij de functie aanroep). Voordat we de prijs in het juiste HTML-veld plaatsen ronden we deze eerst netjes af door de functie 'roundIt' aan te roepen.

- De waarde van het veld 'Total amount ..' moet opgehoogd worden zodat de totale prijs van de bestelling klopt. Dit doen we in regel 21 door simpelweg de totaalprijzen van alle snacks bij elkaar op te tellen.

OPDRACHT 1.4 HET KASSASYSTEEM LATEN WERKEN

In het script zitten een aantal fouten en het script is nog niet compleet. Aan jou de opdracht om het script volledig werkend te krijgen.

1. Zorg dat de case "Hamburger" helemaal goed werkt door de code te verbeteren of aan te vullen. De code is compleet als de 3 punten die hierboven genoemd worden het goed doen.
2. Breidt het Javascript uit en zorg ervoor dat de knoppen voor alle 6 de producten die in de database staan goed werken.
3. Misschien moet er in 'cash_register.php' ook nog wat aangepast worden om te zorgen dat de Javascript code goed uitgevoerd kan worden. Zoek dit uit en pas het aan indien nodig.

THEORIE AFREKENEN

In 'cash_register.php' is er een knop aangemaakt om af te rekenen met de klant. Deze knop doet op dit moment nog helemaal niets, de vorige developer is er niet aan toegekomen om de code hiervoor te schrijven. Het is aan jou om te zorgen dat deze knop wel functionaliteit krijgt. Welke taal/techniek jij hiervoor gaat gebruiken is aan jou, je mag dus gebruik van Javascript, PHP of een combinatie van beide.

Hieronder een overzicht van wat moet er gebeuren op het moment dat er op de knop 'Check out' wordt geklikt:

- Er moet een overzicht van de bestelling getoond worden. Dit mag in de huidige pagina zijn maar het mag ook op een aparte pagina getoond worden. In dit overzicht staat minimaal welke snacks er besteld zijn en de totaalprijs van de gehele bestelling. De layout/vormgeving van dit overzicht mag jij zelf bepalen.
- De ingevulde velden moeten allemaal leeggemaakt worden zodat het kassasysteem klaar staat voor de volgende bestelling.

OPDRACHT 1.5 AFREKENEN

1. Zorg dat de 'Check out' knop werkt zoals hierboven beschreven.

EXTERNE BRONNEN

- [Reader Unity 1: Digitale lesstof/BASIS1B/7. Unity 1](#)

Minor algemeen – opdrachten periode 4 leerjaar 1

- Lesmateriaal PHP: Digitale lesstof/MINOR1/VERDPRO/PHP1
- <https://www.ntchosting.com/encyclopedia/scripting-and-programming/php/php-in/>

OPDRACHT 2 – DE BESTELZUIL (DIGITAL SIGNAGE)

INTRODUCTIE

In elk modern fastfood restaurant is er een mogelijkheid om jouw bestelling via een bestelzuil in te voeren en te betalen. Het restaurant waarvoor je binnen deze module de opdrachten maakt wil ook zo'n bestelzuil aan de klant beschikbaar stellen.



De software voor de bestelzuil laten we weer op een Raspberry Pi draaien, net zoals in de vorige Digital Signage opdracht. Het touchscreen moet nu echter wel in portrait stand gezet worden. De bestelde producten worden in een database op een server opgeslagen.

OPDRACHT

Zoals in de inleiding beschreven staat maak je een bestelzuil voor het restaurant. Je gebruikt dezelfde hardware (Raspberry Pi en touchscreen) voor het realiseren van het product. Het touchscreen moet nu in portrait stand gebruikt worden.

De applicatie bestaat uit drie pagina's:

- Overzichtpagina: Hier worden alle producten getoond. Op deze pagina kan de klant in verticale richting scrollen.
- Bestelpagina: Hier wordt een geselecteerd product getoond en heeft de klant de mogelijkheid om het product in de bestelling op te nemen.
- Afrekenpagina: Hier kan de klant (fictief) afrekenen.

Minor algemeen – opdrachten periode 4 leerjaar 1

De overzichtspagina

De layout van deze pagina moet als volgt worden:



- Zet minimaal 10 producten in de applicatie
- Als er op een bestelknop geklikt wordt springt de applicatie naar de tweede pagina en toont daar het product.

Bij de beschrijving staat uiteraard ook de prijs.

Minor algemeen – opdrachten periode 4 leerjaar 1

De bestelpagina

De layout van deze pagina moet er als volgt worden:

The wireframe shows a rectangular container for the page layout. At the top is a header box labeled 'Header met logo en naam van het restaurant'. Below the header is a large box for the product photo, labeled 'foto van het geselecteerde product'. Underneath the photo is a box for the product description and price, labeled 'beschrijving van het product en de prijs'. Below that is an input field for the quantity, labeled 'Input veld voor het aantal'. At the bottom are two buttons: 'Bestelknop' on the left and 'Afrekenknop' on the right.

- In het inputveld kan de klant het aantal invoeren voor het gekozen product.
- Als er op de bestelknop geklikt wordt, wordt de bestelling naar de server gestuurd en in een database opgeslagen
- Als er op de afrekenknop geklikt wordt springt de applicatie naar de derde pagina

Hoe je de gegevens naar een database moet wegschrijven lees je verderop.

Minor algemeen – opdrachten periode 4 leerjaar 1

De afrekenpagina

De layout van deze pagina moet er als volgt worden:

Header met logo en naam van het restaurant

Bestelnummer en totaalprijs

1 <small>1 1 1</small>	2	3
4 <small>1 1 1</small>	5	6
7 <small>1 1 1</small>	8	9
0 <small>1 1 1</small>	OK	

Eventuele foutmelding

- Als deze pagina getoond wordt, dan verschijnt het bestelnummer en de berekende totaalprijs in beeld
- Als er vier cijfers als pincode ingevoerd zijn kan er op de OK knop geklikt worden. Daarna zal de applicatie terugspringen naar het eerste scherm (bestelling is afgerond)
- Als er geen vier cijfers ingetoetst zijn, maar wel op de OK knip geklikt wordt verschijnt er een foutmelding

Er hoeft geen controle uitgevoerd te worden welke pincode er ingevoerd wordt. Elke combinatie van vier getallen is goedgekeurd. Wel moet er een controle zijn of er vier keer een getal-knop ingedrukt wordt.

EXTERNE BRONNEN

In deze link zie je hoe je een scherm in portrait stand kan zetten op een Raspberry Pi:

<https://www.raspberrypi-spy.co.uk/2017/11/how-to-rotate-the-raspberry-pi-display-output/>

Hoe krijg je een geselecteerd product naar de volgende pagina in javascript:

<https://www.sitepoint.com/get-url-parameters-with-javascript/>

Wat is JSON ?:

https://www.w3schools.com/whatis/whatis_json.asp

In deze link zie je hoe je data naar de server kan sturen:

<https://www.geeksforgeeks.org/how-to-send-a-json-object-to-a-server-using-javascript/>

THEORIE

Opslaan van de gegevens in de database

In deze opdracht gebruiken we een database die bestaat uit één tabel. De tabel ziet er als volgt uit:

id	bestelnummer	productnaam	aantal	prijs	betaald
1	1	Hamburger	2	350	0
2	1	MilkShake	1	200	0
3	2	Friet	3	150	1
4	2	Mayo	2	50	1

Het id is de unieke aanduiding van een rij. Dit is altijd noodzakelijk in een database. De database is zo ingesteld dat het id automatisch door de database wordt ingevuld.

Het bestelnummer geeft aan welke producten er binnen dezelfde bestelling horen. De bovenste twee producten zijn door klant 1 besteld en de volgende twee producten door klant 2.

Het aantal is uiteraard het aantal keer dat het product besteld is. Klant 1 heeft dus 2 hamburgers en 1 milkshake besteld.

De prijs wordt in centen in de database opgeslagen. De hamburger kost dus eigenlijk € 3,50

Als je de totale prijs voor klant 1 gaat berekenen, dan moet je dus 2 keer € 3,50 voor de hamburgers rekenen en 1 keer € 2,00 voor de milkshake (in totaal dus € 9,00)

De database krijg je aangeleverd. De staat in het bestand [bestellingen.sql](#). Hoe je deze op de webserver moet plaatsen zie je in de video: [database_op_plesk.mp4](#)

Om de database te kunnen gebruiken vanuit de door jou te maken javascript code is er een voorbeeldproject aangeleverd. Deze staat in het bestand [bestelzuil.zip](#).

De volgende stappen moet je nemen om het voorbeeldproject werkend te krijgen.

- Maak op de webserver eerst een map aan met de naam [bestelzuil](#).
- Pak het zip-bestand uit en kopieer de inhoud naar de zojuist gemaakte map op de server.

Minor algemeen – opdrachten periode 4 leerjaar 1

- Pas het bestand met de naam config.inc.php aan zoals hieronder aangegeven.

```
<?php
$db_hostname = 'localhost:3306';
$db_username = 'db12345';
$db_password = 'mijnpass';
$db_database = 'bestelzuil';

$mysqli = mysqli_connect($db_hostname, $db_username, $db_password, $db_database);
?>
```

- Verander *db12345* naar de naam die jij ingevoerd hebt bij het aanmaken van de database (zie video [database_op_plesk.mp4](#))
- Doe hetzelfde met *mijnpass*

Het aangeleverde voorbeeldproject bestaat uit vier bestanden. Het bestand config.inc.php dat je net aangepast hebt is bedoeld om vanuit PHP de connectie met de database te maken. Het bestand bestelzuil.php zorgt ervoor dat de gegevens in de database weggeschreven worden. Dit bestand werkt op de server en gaan we vanuit Javascript aanroepen. Je hoeft niets aan te passen in dit PHP bestand.

In het bestand index.html is het javascript bestand gekoppeld en is er een input element in de body geplaatst. Dit input element is een hidden field, waarin we het bestelnummer gaan bijhouden. Deze mag je dus niet verwijderen.

```
<html>
  <head>
    <script src="verzend.js" defer="defer"></script>
  </head>
  <body>
    <input type="hidden" id="bestelnummer" value="0">
  </body>
</html>
```

In het javascript document staan twee functies en een stuk testcode. De eerste functie wordt gebruikt om een bestelling naar de server te sturen en via het PHP bestand in de database weg te schrijven.

```
function zendBestelling(url, bestelnummer, product, aantal, prijs){
  // Creating a XHR object
  let xhr = new XMLHttpRequest();

  // open a connection
  xhr.open("POST", url, true);

  // Set the request header i.e. which type of content you are sending
  xhr.setRequestHeader("Content-Type", "application/json");

  // Create a state change callback
  xhr.onreadystatechange = function () {
    if (xhr.readyState === 4 && xhr.status === 200) {
      // Geef het bestelnummer als antwoord terug;
      document.getElementById("bestelnummer").value = this.responseText;
    }
  };

  // Converting JSON data to string
  var data = JSON.stringify({ "bestelnummer": bestelnummer, "product": product ,
    "aantal": aantal, "prijs": prijs});

  // Sending data with the request
  xhr.send(data);
}
```

De functie heeft als input de url naar de PHP pagina, het nummer van de bestelling, de naam van het product, het aantal en de prijs. Als het bestelnummer de waarde 0 heeft, betekend dit dat er een nieuwe klant is en wordt door het PHP bestand het juiste bestelnummer bepaald en teruggestuurd naar deze functie. In deze functie wordt het hidden input veld geüpdatet met het juiste bestelnummer. Een volgende bestelling door dezelfde klant kan dus met hetzelfde bestelnummer uitgevoerd worden (Dit zie je straks in de testcode).

De tweede functie is voor het afronden van de bestelling. Als op pagina 3 op de OK knop geklikt wordt (nadat er een pincode is ingevoerd) wordt deze functie aangeroepen. Hierbij wordt enkel de url en het bestelnummer doorgegeven:

```
function zendBetaling(url, bestelnummer){
    // Creating a XHR object
    let xhr = new XMLHttpRequest();

    // open a connection
    xhr.open("POST", url, true);

    // Set the request header i.e. which type of content you are sending
    xhr.setRequestHeader("Content-Type", "application/json");

    // Create a state change callback
    xhr.onreadystatechange = function () {
        if (xhr.readyState === 4 && xhr.status === 200) {
            // Geef het bestelnummer als antwoord terug;
            document.getElementById("bestelnummer").value = "0";
        }
    };

    // Converting JSON data to string
    var data = JSON.stringify({ "bestelnummer": bestelnummer });

    // Sending data with the request
    xhr.send(data);
}
```

Als deze functie aangeroepen wordt, dan zal er weer contact gemaakt worden met de PHP code. In de PHP code wordt de bestelling op betaald gezet. In de laatste kolom van de tabel wordt de waarde 0 veranderd in de waarde 1. Ook zorgt de functie ervoor dat de waarde van het hidden inputveld gereset wordt naar 0, zodat een volgende klant een nieuw bestelnummer krijgt als er een bestelling gedaan wordt.

In de testcode zie je dat de url naar het php bestand is opgenomen en dat er twee producten besteld worden en tot slot de betaling wordt doorgevoerd. In de code staat aangegeven dat er elke keer een korte wachttijd tussen zit. Dit is nodig omdat de communicatie met de server tijd kost en deze code direct na elkaar wordt uitgevoerd. In jouw code hoeft je daar geen rekening mee te houden.


```
//de url waar het php bestand staat
let url = "bestelzuil.php"

//eerste bestelde product
let product = "Hamburger";
let aantal = 2;
let prijs = 350;    //prijs in centen

//aanroep van de functie
let bestelnummer = document.getElementById("bestelnummer").value;
zendBestelling(url, bestelnummer, product, aantal, prijs);

//Na 2 seconde wordt een tweede product besteld
//met hetzelfde bestelnummer (dus hoort het bij dezelfde bestelling)
setTimeout(() => {
    bestelnummer = document.getElementById("bestelnummer").value;
    product = "kipnuggets";
    aantal = 1;
    prijs = 300;

    zendBestelling(url, bestelnummer, product, aantal, prijs);
}, 2000);

//na weer twee seconde wordt de bestelling betaald
setTimeout(() => {
    bestelnummer = document.getElementById("bestelnummer").value;
    zendBetaling(url, bestelnummer);
}, 4000);
```

Je ziet dat bij elke aanroep van één van de functies altijd eerst het hidden field wordt uitgelezen omhet juiste bestelnummer mee te sturen.

Als je zelf aan de slag gaat kan je deze testcode verwijderen met uitzondering van de url.

Data doorgeven tussen de verschillende pagina's

Het is ook nodig om data door te geven tussen de verschillende pagina's. Stel je klikt op de eerste pagina op de hamburger, je kan hiervoor de volgende link gebruiken:

```
<a href="pagina2.html/?product=hamburger&prijs=300">...</a>
```

Achter de url zet je dus extra variabelen die je meestuurt. In dit voorbeeld is dat het product en de prijs. Op pagina2 kan je deze informatie eruit halen. Hoe je dit kan doen zie in de volgende link:

<https://www.sitepoint.com/get-url-parameters-with-javascript/>

OPDRACHT 3 – KLANTEN-OPROEPSYSTEEM (IOT)

INTRODUCTIE

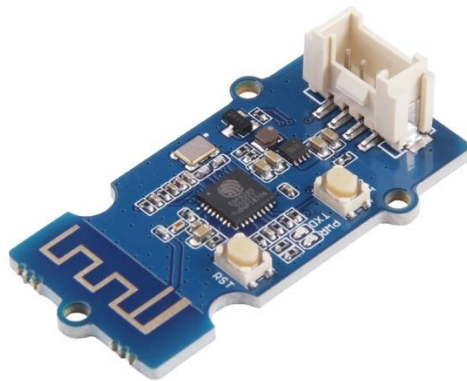
In sommige snackbars krijg je wanneer je besteld hebt een zogenaamde *pager* of *coaster* mee: een apparaat dat gaat piepen wanneer je bestelling klaar is. Dit heet een klanten-oproepsysteem.

Voor ons restaurant gaan we dit systeem met behulp van Arduino's zelf maken. De klant krijgt bij de bestelling onze zelfgemaakte pager mee. Als de bestelling klaar is, wordt deze vanuit het backend systeem opgeroepen en gaat die piepen. Dan weet de klant dat de bestelling klaar is en kan worden opgehaald.



OPDRACHT 1: WIFI

Om te beginnen moeten we onze Arduino met het internet verbinden. Daarvoor hebben we een WiFi adapter nodig. We gebruiken de **Grove – UART WiFi V2**, die aansluit op het Grove systeem dat we gebruiken.



Op de website van de fabrikant (Seeed) kun je op [deze pagina](#) meer lezen over de module die we gebruiken. De module wordt bestuurd door een ESP8266 chip. Dit is in feite zelf ook weer een microcontroller, net als de Arduino. We gaan niet de 'Getting started' stappen van die pagina volgen. Daar werken ze namelijk met een net iets ander board: de Seeduino Lite. Wij werken met de Arduino Uno.

1.1 BAUDRATE VERANDEREN

Om hem aan te sluiten hebben we direct al een probleem: de WiFi module moet worden aangesloten met een zogenaamde *Serial* (seriële) poort. Daar heeft onze Arduino Uno er maar één van en die gebruiken we al om de Arduino te verbinden met onze computer.

Er is een oplossing, we kunnen op de arduino een software-matige serial poort gebruiken met behulp van de *SoftwareSerial* library. Maar het volgende probleem dat we daarbij tegenkomen is dat deze SoftwareSerial poort alleen goed werkt met verbindingen met een *baudrate* (overdrachtsnelheid) tot 9600, terwijl de WiFi module standaard staat ingesteld om met een baudrate van 115200 te werken.

Het eerste wat we dus moeten doen is de baudrate van de WiFi module veranderen naar 9600. Dat doen we met de Arduino.

1. Plug de WiFi adapter met een Grove kabeltje in de port D6 op het Grove shield.
2. Verbind je arduino met de computer, open de Arduino IDE en selecteer (zodanig) de juiste poort in de Arduino IDE.
3. Open de Serial monitor en stel hem in op 9600 baud
4. Open een nieuwe sketch, plak onderstaande code in de sketch en upload het naar je Arduino:

```
/*  
Change the baudrate on a connected ESP8266 module  
*/  
  
#include <SoftwareSerial.h>  
  
SoftwareSerial mySerial(6, 7); // RX, TX  
  
long int currentBaudRate = 115200;  
long int newBaudRate = 9600;  
  
void setup() {  
  // Open serial communications and wait for port to open:  
  Serial.begin(115200);  
  
  Serial.println("Started.");  
  
  // set the data rate for the SoftwareSerial port  
  mySerial.begin(currentBaudRate);  
  
  // send the command to change the baud rate  
  mySerial.print("AT+UART_DEF=" + String(newBaudRate) + ",8,1,0,0\r\n");  
  mySerial.flush();  
  mySerial.begin(newBaudRate);  
  
  Serial.println("Changed.");  
}
```

```
delay(2000);  
// send the command to test  
mySerial.print("AT+GMR\r\n");  
}  
  
void loop() {  
// send all incoming data from the ESP to the serial port to monitor  
if (mySerial.available()) {  
  Serial.write(mySerial.read());  
}  
}
```

In dit script openen we twee seriële poorten: een gewone seriële poort met de computer om informatie te kunnen loggen (met `Serial.begin(9600)`) en een SoftwareSerial poort (genaamd `mySerial`) om met de WiFi adapter te communiceren.

Dan sturen we het commando “AT+UART_DEF...” naar de WiFi adapter. Dit is het commando om de baudrate te veranderen. Met zogenaamde “AT” commands kunnen we opdrachten naar de WiFi adapter sturen.

Als het goed is gegaan, zie je na een korte tijd de volgende tekst in de monitor verschijnen:

```
Started.  
Changed.  
AT+UART_DEF=9600,8,1,0,0  
  
OK  
AT+GMR  
AT version:1.6.0.0(Feb 3 2018 12:00:06)  
SDK version:2.2.1(f42c330)  
compile time:Feb 12 2018 16:31:26  
Bin version(Wroom 02):1.6.1  
OK
```

1.2 EEN HTTP REQUEST DOEN

In de volgende stap gaan we met de Arduino een pagina op het internet openen.

Om het onszelf gemakkelijk te maken gaan we een library gebruiken die de WiFi module aanspreekt, zodat we niet zelf AT commands te hoeven gebruiken. Deze library heet [WiFiEsp](#). Installeer deze library door de code van github als zip te downloaden en in de Arduino IDE de zip in te laden. Voor hulp hoe dat gaat kun je de beschrijving op [deze pagina](#) gebruiken.

Je hebt dan onder Examples in het menu een nieuw lijstje erbij genaamd “WiFiEsp”. Open het WebClientRepeating voorbeeld hiervan, pas de volgende drie dingen aan:

- Het WiFi ssid
- Het WiFi wachtwoord

- De baudrate van de gewone Serial verbinding met de computer stel je in op 9600

Dit laatste doe je door in de setup functie

```
Serial.begin(115200);
```

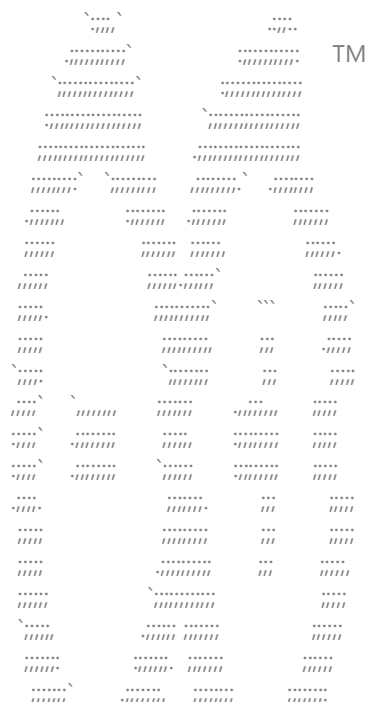
Te vervangen met

```
Serial.begin(9600);
```

Upload je sketch naar je Arduino en bekijk de Serial monitor. Als het goed gaat zie je het volgende verschijnen:

```
[WiFiEsp] Initializing ESP module
[WiFiEsp] Initialization successful - 2.2.1
Attempting to connect to WPA SSID: ###
[WiFiEsp] Connected to ###
You're connected to the network
SSID: ###
IP Address: 192.168.68.120
Signal strength (RSSI):-542 dBm

[WiFiEsp] Connecting to arduino.cc
Connecting...
HTTP/1.1 200 OK
Server: nginx/1.4.2
Date: Tue, 18 May 2021 09:54:14 GMT
Content-Type: text/plain
Content-Length: 2263
Last-Modified: Wed, 02 Oct 2013 13:46:47 GMT
Connection: close
Vary: Accept-Encoding
ETag: "524c23c7-8d7"
Accept-Ranges: bytes
```



The figure consists of a 4x4 grid of 16 small diagrams. Each diagram shows a 2D lattice of particles, represented by dots. The diagrams illustrate the evolution of a wave-like pattern of particles over time. The top row shows a single wave pulse. The second row shows the pulse interacting with a boundary. The third row shows the pulse reflecting. The bottom row shows the pulse continuing its motion.

```

***          *****          **          ****          ***
** 11 11111  *****  *11 11 11111 11* 11 1111
***          *****          **          ****          ***
** 11-111  *****  *11 11 111111 11 11 11-11
11*1  ** 11  ** 11 *11 11 11 1111 11 11 11 11
** 11 11 11 11 11 11 11 11 11 11 11 11 11 11
** 11 * 11 11 11 11 *11 11 111 111111 11 11 11*
1* 11 ***** 11 11 *11 11 11 11 11 11 11 11 11
***** ** 11 11 11 11 11 11 11 11 11 11 11 11
***** 11 11 11 11 *11 11 11 11 11 11 11 11 11
** 11 11 ***** ***** ** *****
** 11 11 *11 111111* 111111* 11 111 111111
** 11 11 ***** 111111 ***** ** 11 *****
11 11 11 11 1111* 1111* 111111 11 11 1111

```

Vervolgens zal de Arduino elke 10 seconden deze pagina herladen.

OPDRACHT 2: DE PAGER

In principe hebben we hiermee genoeg bouwstenen om onze pager te maken. Wat we vervolgens moeten doen:

1. In ons backend systeem een pagina maken die de Arduino elke 10 seconden opvraagt. In die pagina staat bijvoorbeeld alleen het woord: " bezig ". Op het moment dat de bestelling klaar is moet die pagina het woord " klaar " laten zien.
2. De Arduino moet deze pagina elke 10 seconden openen.
3. De Arduino moet het woord " klaar " herkennen.
4. De Arduino moet dan een geluidsignaal afgeven.

2.1 DE PAGINA

Dit kan een simpel php script zijn dat in de database uitleest of de order klaar is of niet.

Bijvoorbeeld:

```
<?php

// check in de database of de order klaar is en sla dat op in variabele $klaar...

if ($klaar) {
    echo "klaar";
} else {
    echo "bezig";
}
```

We hoeven geen html opmaak mee te sturen, want de pagina wordt alleen door de Arduino opgevraagd.

2.2 DE PAGINA OPVRAGEN MET DE ARDUINO

Begin met het *WebClientRepeating* voorbeeld en werk vanuit dat script.

In onderstaande voorbeeld wordt uitgegaan van het adres <https://bijster.ict-lab.nl/iot/klaar.php>. Pas de hostnaam en pad naar de pagina aan aan jouw situatie. (Je kunt ook eerst deze voorbeeldpagina gebruiken om te testen, deze pagina geeft altijd alleen het woord "klaar" terug. Er is ook een [bezig.php](#) die "bezig" teruggeeft)

Om te beginnen moeten we onze pagina opvragen in plaats van het tekstbestand van arduino.cc dat nu wordt opgevraagd. Als we zouden werken met een onbeveiligd *http request*, dan stellen we dat als volgt in:

Nog voor de `setup()` functie, stel je de servernaam in:

```
char server[] = "bijster.ict-lab.nl";
```

Vervolgens, in de `httpRequest()` functie, stel je het pad in van de pagina op de server en nogmaals de host

```
...
client.println(F("GET /iot/klaar.php HTTP/1.1"));
client.println(F("Host: bijster.ict-lab.nl"));
...
```

Stel dit in en bekijk wat er gebeurt. Wat zie je?

Als het goed is, zie je nu dat de server een *301* code teruggeeft. Wat er gebeurt is het volgende: de server van school staat ingesteld om requests automatisch door te sturen naar een beveiligde *https* verbinding. We moeten dus een beveiligde verbinding openen om de pagina te zien. Dat doe je door in de `httpRequest()` functie deze regel

```
if (client.connect(server, 80)) {
```

te veranderen in:

```
if (client.connectSSL(server, 443)) {
```

Als je dit verandert en de sketch nogmaals upload zou je moeten zien dat je het bericht "klaar" (of "bezig") terugkrijgt.

2.3 HET WOORD "KLAAR" HERKENNEN

Hoe zien we of het woord "klaar" in de response terugkomt?

In de functie `loop()` is een *while* loop waarin alles wat we terugkrijgen van de server letter voor letter naar de seriële poort met de computer wordt geschreven, zodat we in de serial monitor kunnen zien wat er terugkomt.

Deze letters kunnen we dus ook steeds toevoegen aan een string. Als die string dan het woord "klaar" bevat is het gerecht klaar.

Om te beginnen moeten we deze string globaal definiëren. Typ bovenin je applicatie, **nog voor** de `setup()` functie:

```
String response = "";
```

Pas dan de code in de `while` loop in de `loop()` functie het volgende aan:

```
...
void loop()
{
  // if there's incoming data from the net connection send it out the serial port
  // this is for debugging purposes only
  while (client.available()) {
    char c = client.read();
    Serial.write(c);

    // de volgende code voegen we toe:
    // voeg de nieuwe letter bij de response string
    response += c;
    // eindigt de response tot nu toe op "klaar"?
    if (response.endsWith("klaar")) {
      // Ja!
      // Zelf doen: zorg ervoor dat het geluidssignaal afgaat
      // Om te checken schrijven we dit ook naar de seriele poort
      Serial.write(" ### Klaar! Bzzzz ### ");
      // En we stellen de response weer opnieuw in
      response = "";
    }
  }
}
...
```

Maar als het gerecht nog niet klaar is, zal deze response string steeds blijven groeien. Het geheugen op een Arduino is beperkt, dus voor de zekerheid maken we deze string weer leeg elke keer voordat we een nieuw request doen. In het begin van de `httpRequest()` functie schrijf je:

```
void httpRequest()
{
  response = "";
  ...
}
```

Als dat werkt, zie je het volgende in de serial monitor:

```
[WiFiEsp] Connecting to bijster.ict-lab.nl
Connecting...
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 18 May 2021 11:57:18 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/7.3.28
X-Powered-By: PleskLin
```

```
5
klaar ### Klaar! Bzzzz ###
0
```

2.4 EEN GELUIDSSIGNAAL AFGEVEN

In de Arduino dozen zit onder andere een speakertje dat je op de Arduino kunt aansluiten; de [Grove Buzzer](#). Zoek op hoe je dit element werkend krijgt en zorg dat de buzzer afgaat wanneer de bestelling klaar is.

EXTRA: MEERDERE PAGERS

Stel dat je meerdere pagers tegelijkertijd wil gebruiken. Dan moeten ze niet allemaal afgaan wanneer één bepaalde bestelling klaar is.

Wat je dan zou kunnen doen, is een *GET* parameter met het pager nummer meegeven aan de pagina die je opvraagt. Dit kun je dan in php uitlezen en dan kijken of de order die hoort bij dit pager nummer klaar is. Zorg voor extra punten ervoor dat je bovenin het Arduino script een pager nummer in kan stellen en dat die wordt meegestuurd als GET parameter aan het request. Laat zien dat het resultaat (of die klaar is of niet) anders is met een ander pager nummer.

EXTERNE BRONNEN

- https://wiki.seeedstudio.com/Getting_Started_with_Arduino/
- https://wiki.seeedstudio.com/Grove-UART_Wifi_V2/
- <https://github.com/bportaluri/WiFiEsp>
- <https://wiki.seeedstudio.com/Grove-Buzzer/>

OPDRACHT 4 – HAMBURGER OP TAFEL MAKEN (XR)

INTRODUCTIE

In deze module gaan we aan de slag met AR (Augmented Reality). We bouwen verder op de vorige AR module. We gaan in deze module uit van het gebruik van een Android toestel om het project te builden en te testen.

Er wordt in deze module een project opgezet waar je met de telefoon camerabeelden kan herkennen en hier een virtueel object op kan projecteren. Dit doen we met ARFoundation in Unity. De eerste les gaan we de basis bekijken en in de tweede les gaan we aanpassingen op de standaard maken om er ons eigen project van te maken.

EXTERNE BRONNEN

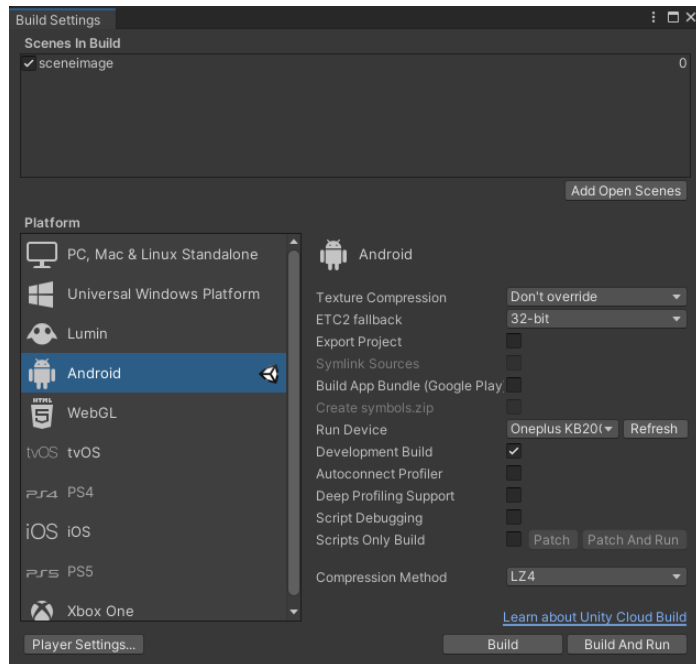
<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html>
<https://docs.unity3d.com/Manual/index.html>
<https://assetstore.unity.com/>
<https://learn.unity.com/tutorial/setting-up-ar-foundation#5fe2be51edbc2a1f5e69872f>
<https://forum.unity.com/threads/ar-multiple-image-tracking-multiple-objects.846901/>
[Unity AR](#)



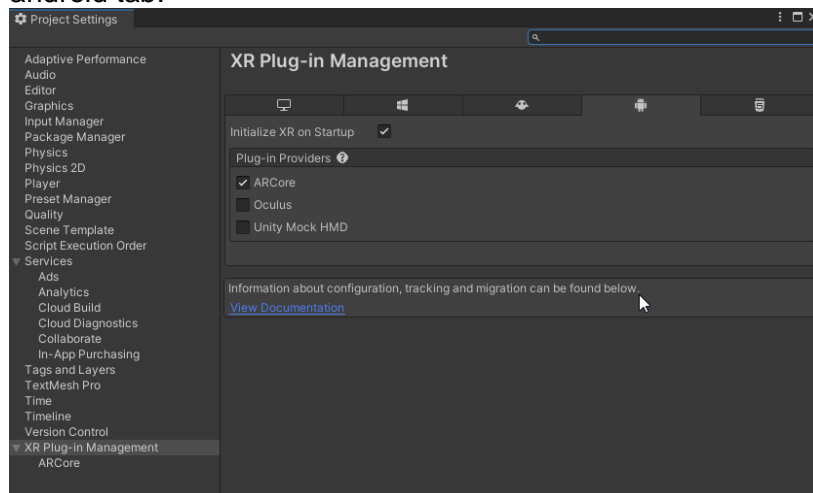
THEORIE

LES 1, SETUP GENERIEKE IMAGE TRACKING

Open een nieuw Unity project en zorg ervoor dat AR Foundation in het project staat. Om dit te checken kan je naar Window > Package Manager hier zou AR Foundation moeten staan. Indien dat niet het geval is kan je naar deze site <https://learn.unity.com/tutorial/setting-up-ar-foundation#5fe2be51edbc2a1f5e69872f>

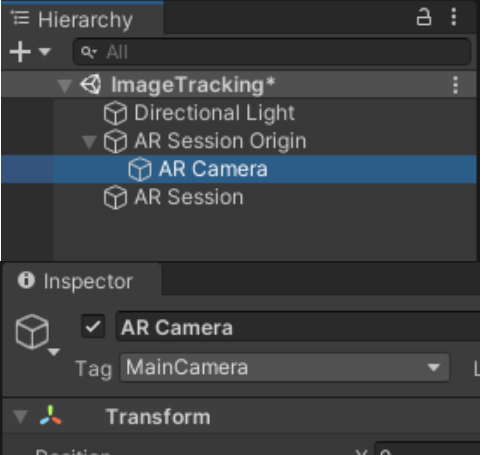
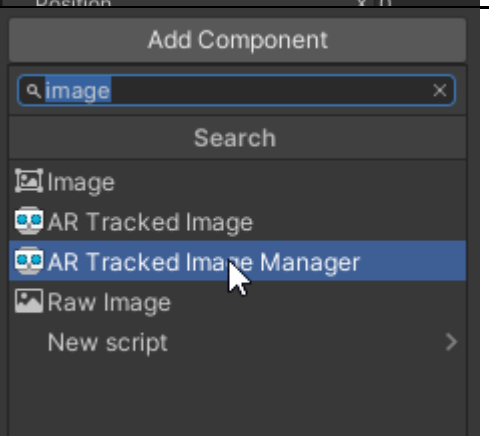


Vergeet niet om de build settings aan te passen naar een Android device. File > Build Settings en in de Edit > Project Settings de checkbox **ARCore** aan te vinken onder de android tab.



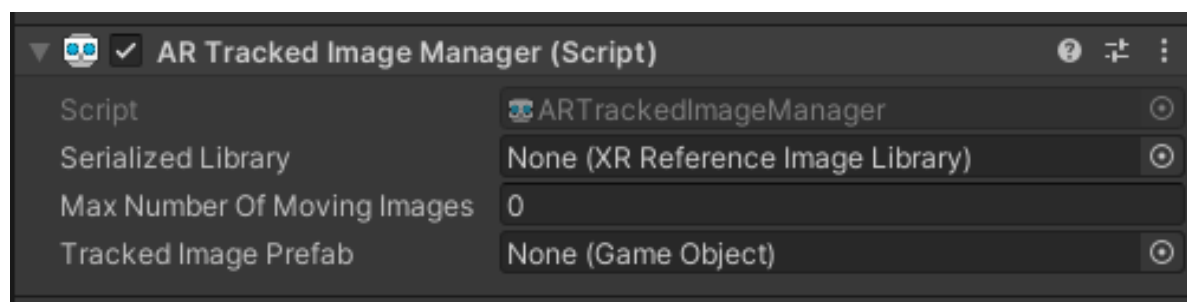
Maak een nieuwe Scene aan. Verwijder de Main Camera. Nu hebben we een “clean” setup voor het maken van een image tracking AR.

Als eerste willen we twee XR objecten toevoegen. Een AR Session Origin (RMB in de Scene > XR > AR Sessions Origin) en een AR Session (RMB in de Scene > XR > AR Session). De AR Session Origin is het object wat ervoor gaat zorgen dat de AR op onze telefoon gaat draaien. AR Session zorgt ervoor dat alle mogelijkheden van onze Unity ARCore bereikbaar zijn binnen het project.

<p>Binnen AR Session Origin zit een AR Camera. De AR Sessions Origin kan opengeklapt worden in de Scene Hierarchy. Selecteer de AR Camera om de inspector van dit element te bekijken. Bovenin de Inspector kunnen we een Tag toekennen. Deze willen we veranderen van “Untagged” naar “Main Camera”. Op deze manier weet het project waar het beeld vandaan moet komen bij het opstarten.</p>	
<p>Nu willen we de image tracking gaan toevoegen. Selecteer de AR Session Origin en klik op de Add Component knop in de Inspector. Typ in “Image” en Selecteer <u>AR Tracked Image Manager</u>.</p>	

Het resultaat moet er ongeveer uitzien als de afbeelding hieronder. We hebben hier 4 items aan het script hangen;

- Script
 - Het script zelf
- Serialized Library
 - Een folder aan images die “getracked” gaan worden.
- Max Number of moving Images
 - De maximale hoeveelheid aan objecten die we over images heen kunnen plaatsen om het werkgeheugen in de gaten te houden.
- Tracked Image Prefab
 - Het object wat op de images geplaatst wordt door middel van AR.

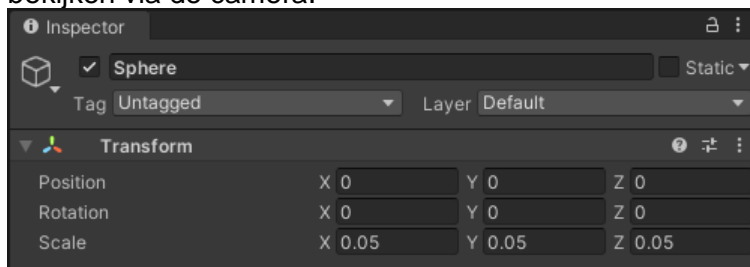


Om die images toe te kunnen voegen aan de Serialized Library hebben we een plek nodig om die images vandaan te halen. AR Foundation heeft hier al over nagedacht. We kunnen

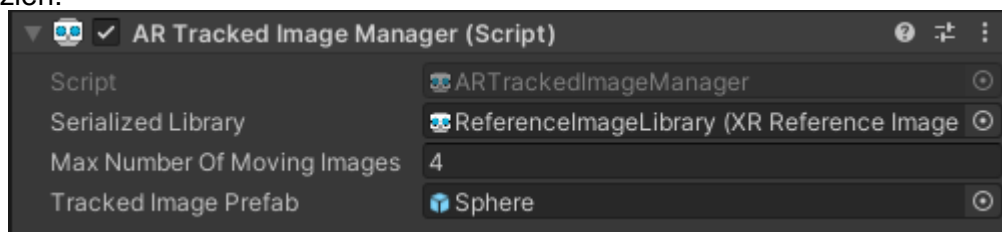
hier een nieuwe Prefab voor aanmaken. Ga naar de Prefab folder (maak deze aan als je deze nog niet hebt). In de Prefab folder: Rechtermuisknop > Create > XR > Reference Image Library. Nu hebben we een nieuwe Prefab voor het verzamelen van images die getracked moeten worden. Als je de Prefab selecteert kun je in de Inspector zien dat hier simpel Images aan toegevoegd kunnen worden. Zorg er wel voor dat de images die je toevoegt al in je project staan, anders kan Unity ze niet vinden. Dit moet een .jpg bestand zijn.

Zodra de image library gevuld is met een of meer foto's kunnen we deze toe gaan voegen aan de AR Session Origin. Klik weer op de AR Session Origin om de inspector te openen. In het toegevoegde script (AR Tracked Image Manager) staat de Serialized Library. Drag and drop de Reference Image Library hier in. Aangezien we een Object willen toevoegen willen we de Max number of moving images op 1 of hoger zetten om het te testen.

Als laatste hebben we een object nodig wat we kunnen inladen op het moment dat een van de images uit de library wordt herkend. Voor nu gaan we een simpele bal (sphere) gebruiken. Om het uit te proberen. In de Scene Hierarchy; RMB > 3D object > Sphere. Voor de AR Camera is het van belang dat we goed kijken naar de grootte van een object. Veel objecten in Unity zijn al snel te groot voor deze camera om goed op te kunnen pakken. We willen van deze bal weer een Prefab maken om te kunnen hergebruiken in de code. Om de bal bruikbaar te maken klikken we het object aan om naar de inspector te gaan. Hier willen we kijken naar de "Transform" kop. Zorg ervoor dat alle Positions op 0 staan. Voor mijn test heb ik de Scales op 0.05 gezet. Dit zorgde ervoor dat de bal klein genoeg was om te bekijken via de camera.



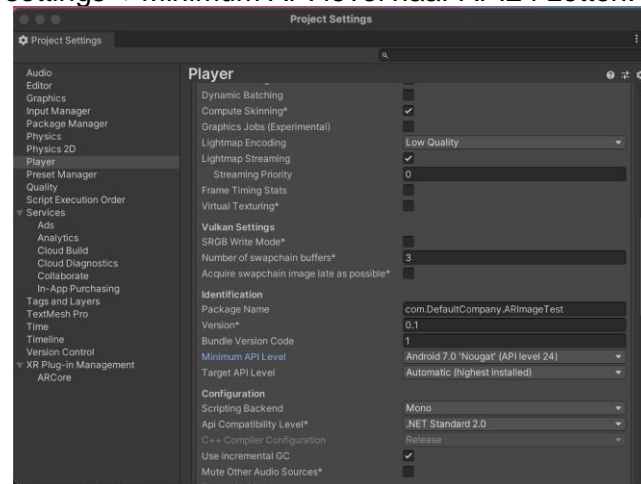
Zodra de Sphere de juiste afmetingen heeft kan hij vanuit de Hierarchy naar de Prefab folder geslept worden. Dit maakt automatisch een Prefab aan. Als de Sphere in de folder staat kan hij uit de Scene gehaald worden. Het laatste wat ons nu nog te doen staat is de Prefab-Sphere te drag-and-droppen in het laatste invulveld van de AR Session Origin > AR Tracked Image Manager, de Tracked Image Prefab. Het resultaat zou er ongeveer zo uit moeten zien:



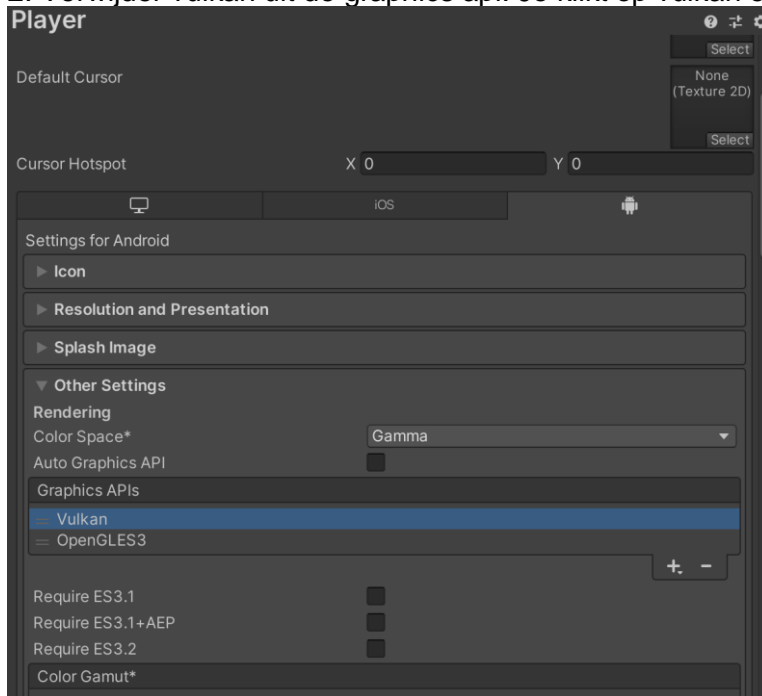
Nu is er een simpele versie van Image Tracking gemaakt en zou het moeten werken. Voordat we gaan builden moeten twee nog 3 kleine taken uitvoeren.

Minor algemeen – opdrachten periode 4 leerjaar 1

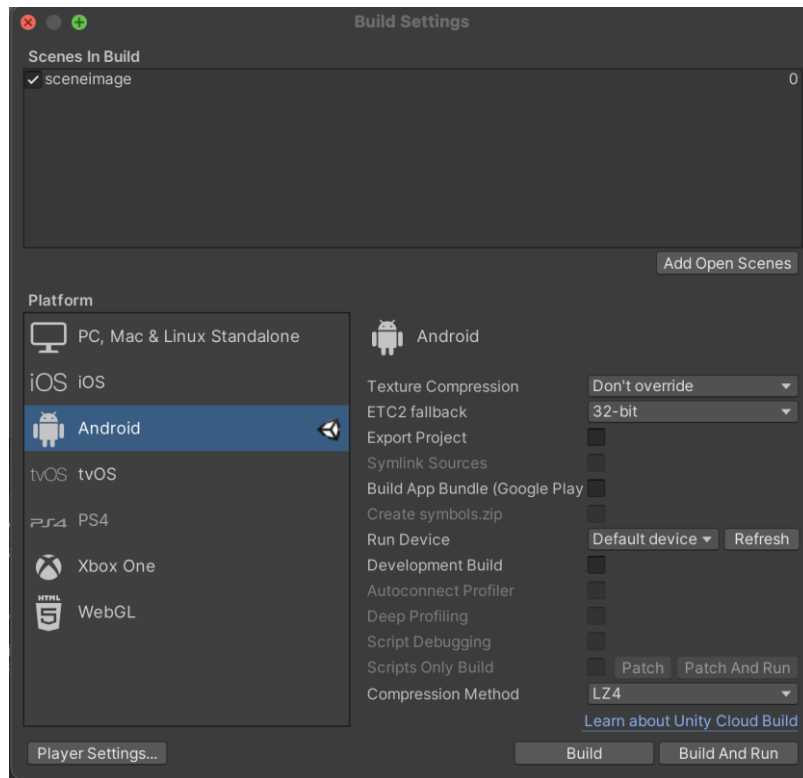
1. De SDK versie moet op versie 24 gezet worden. Dat doe je bij player settings -> other settings -> Minimum API level naar API24 zetten.



2. Verwijder vulkan uit de graphics api. Je klikt op vulkan en dan op het minnetje.



Als laatste moeten we de scene aan de build index toevoegen. Dit doe je via File -> Build settings -> Add open scene. Dan staat de scene daadwerkelijk in de build.



Build en Run het project en kijk of het allemaal werkt!

Mogelijk probleem met builden:

Probleem: USB/mobiel niet gevonden

Oplossing 1: Ga naar ontwikkelingsoptie en klik een paar keer op USB rechten intrekken. Kabel uit de telefoon. USB debuggen uit en aanzetten. Kabel weer in telefoon stoppen. Herhalen totdat je de melding krijgt om iMac te vertrouwen.

Oplossing 2: Zet je telefoon op “bestanden overzetten tijdens opladen”.

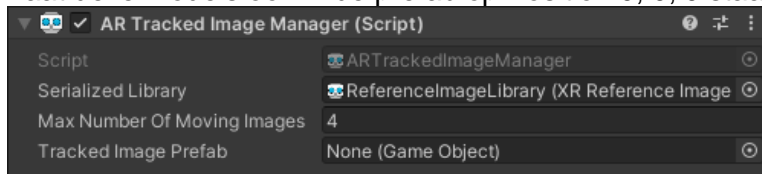
Oplossing 3: Zet kabel connectie automatisch op bestanden overzetten via developer optie.

Zie video: <https://youtu.be/LID8mVZGzs4>

LES 2, MULTIPLE OBJECTS IN IMAGE TRACKING

In deze les gaan we verder met het project van de voorgaande les. We gaan verder kijken naar het tracken van images, maar we gaan dit dusdanig aanpassen zodat er meerdere models in te laden zijn a.d.h.v. verschillende images.

Zet alvast een aantal duidelijke en verschillende images klaar en zorg ervoor dat je een paar models hebt staan die ongeveer dezelfde grootte hebben als de Sphere van de vorige les. Laat deze models ook in de prefab op Position 0, 0, 0 staan.



Als eerste willen we het vorige project klaarzetten om onze eigen code toe te gaan voegen. Hiervoor willen we terug naar de AR Session Origin > AR Tracked Image Manager (Script). Hier willen we de laatste optie - Tracked Image Prefab - leeg halen. We doen dit omdat we zelf prefabs in gaan laden terwijl het programma draait. Als deze optie niet leeg is zal dit boven op onze eigen code ook geplaatst worden.

Nu gaan we aan het echte werk; het maken van een eigen script. Ga naar de (of maak een) Scripts folder. Rechtermuisknop > Create > C# Script, geef een logische naam aan het script bijvoorbeeld "ImageTracking".

Open het nieuw aangemaakte script in een editor. Het eerste wat we willen doen is de standaard logica verwijderen tot er alleen nog de libraries en de class in staan.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class ImageTracking : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
19
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class ImageTracking : MonoBehaviour
6 {
7
8 }
9
```

Nu willen we alle libraries toe gaan voegen waar we gebruik van maken. We willen de Linq, Unity Events, AR Foundation en AR Subsystems gaan gebruiken.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Linq;
4  using UnityEngine;
5  using UnityEngine.Events;
6  using UnityEngine.XR.ARFoundation;
7  using UnityEngine.XR.ARSubsystems;
8
9  public class ImageTracking : MonoBehaviour
10 {
```

Vanaf dit punt gaat alle code die we toevoegen binnen de class ImageTracking komen. We willen starten met 2 globale variabelen; een string om de naam van de image op te halen en de ARTrackedImageManager. De ARTrackedImageManager is een instantie van een vooraf gedefinieerde class uit het AR pakket dat we gebruiken. Tijdens het schrijven van de code gebruiken we dit object om de functies van het AR pakket te benutten.

```
8  public class ImageTracking : MonoBehaviour
9  {
10     public string ReferenceImageName;
11     private ARTrackedImageManager _TrackedImageManager;
12 }
```

Nu gaan we een Awake functie aanmaken. Let op dat de benaming van deze functie exact moet kloppen, ook de hoofdletter. Dit is een functie waar Unity gebruik van maakt. Zodra het "Awake" event wordt aangeroepen zal onze code uitgevoerd worden. Zoals de naam al doet denken; bij het opstarten van het script zal dit event afgetrapt worden en wordt de code uitgevoerd. Bij het opstarten willen we de ARTrackedImageManager opzoeken en in de globale variabele stoppen zodat we hiermee kunnen communiceren in de rest van het script.

```
13     private void Awake()
14     {
15         _TrackedImageManager = FindObjectOfType<ARTrackedImageManager>();
16     }
```

Net zoals de Awake functie gaan we gebruik maken van de OnEnable en OnDisable functies. Ook dit zijn Unity eigen functies, dus let op de naamgeving. In beide gevallen willen we een check doen om te kijken of we een instantie van de tracked image manager hebben. Indien deze niet bestaat kunnen we niet verder. Bij de OnEnable functie willen we data toevoegen aan de trackedImageManager, dit doen we bij de trackedImagesChanged. Bij de OnDisable gaan we deze data juist weghalen. Dit gebeurt respectievelijk door de += en -= operators. De data die we gaan gebruiken gaan we in een volgende functie definiëren. Voor nu gebruiken we alleen de naam van de functie OnTrackedImagesChanged.

```

18     private void OnEnable()
19     {
20         if (_TrackedImageManager != null)
21         {
22             _TrackedImageManager.trackedImagesChanged += OnTrackedImagesChanged;
23         }
24     }
25
26     private void OnDisable()
27     {
28         if (_TrackedImageManager != null)
29         {
30             _TrackedImageManager.trackedImagesChanged -= OnTrackedImagesChanged;
31         }
32     }

```

In deze functie wordt een parameter meegegeven; de `ARTrackedImagesChangedEventArgs`. Deze parameter houdt bij welke images er zijn ontdekt aangepast of weggehaald (added, updated en removed). Er staat een stuk debugging om in de console te kunnen zien wat er binnenkomt via deze parameter. Daarna wordt een nieuwe functie aangeroepen met de toegevoegde en aangepaste images. Dit zijn de images die ontdekt zijn of verplaatst zijn. Daar willen we het object op gaan plaatsen wat we doen in de volgende functie.

```

34     private void OnTrackedImagesChanged(ARTrackedImagesChangedEventArgs e)
35     {
36         foreach (var trackedImage in e.added)
37         {
38             Debug.Log($"Tracked image detected: {trackedImage.referenceImage.name} with size: {trackedImage.size}");
39         }
40
41         UpdateTrackedImages(e.added);
42         UpdateTrackedImages(e.updated);
43     }

```

Nu is het de bedoeling om de functie te maken waar we bij de vorige snippet zijn gebleven. `UpdateTrackedImages`, we geven hier weer een parameter aan mee, dit wordt een `IEnumerable` van het object `ARTrackedImage`. Vanuit de parameter willen we een specifieke image op gaan halen. Dit doen we met de `FirstOrDefault` functie. Hier gaan we het eerste object ophalen wat aan de eisen voldoet. Hier kijken we of de naam van de reference image hetzelfde is als de image die we hebben kunnen tracken. Indien dit niet gelukt is gebruiken we een `return` om de rest van de functie af te sluiten.

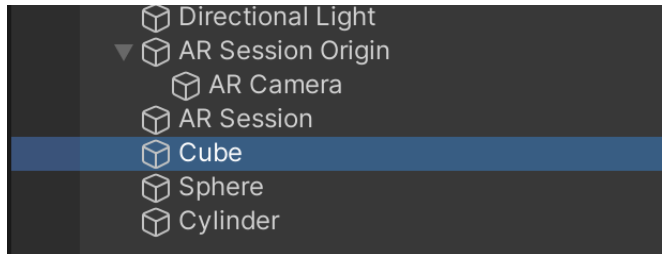
```

45     private void UpdateTrackedImages(IEnumerable<ARTrackedImage> trackedImages)
46     {
47         // If the same image (ReferenceImageName)
48         var trackedImage = trackedImages.FirstOrDefault(x => x.referenceImage.name == ReferenceImageName);
49         if (trackedImage == null)
50         {
51             return;
52         }
53     }

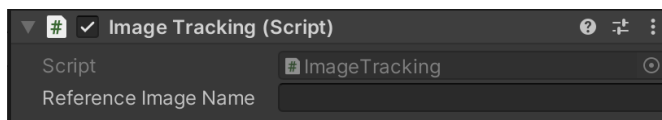
```

Nu weten we zeker dat we een image hebben die overeenkomt met onze reference. Alles wat we nu nog hoeven te doen is het object plaatsen op de plek van de gevonden image. Om dat te kunnen doen moeten we kijken naar de `trackingState` van de image. We moeten genoeg informatie van de image hebben om deze te kunnen gebruiken. Zodra we de check hebben gemaakt gaan we de positie ophalen en gebruiken op het object dat we willen plaatsen. Dit gebeurt met de functie `SetPositionAndRotation`. Hier geven we zowel de position als de rotation aan mee om het object op de juiste manier op de image te plaatsen.

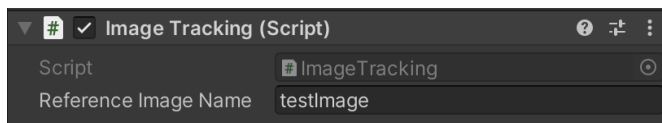

```
52     }
53
54     if (trackedImage.trackingState != TrackingState.None)
55     {
56         var trackedImageTransform = trackedImage.transform;
57         transform.SetPositionAndRotation(trackedImageTransform.position, trackedImageTransform.rotation);
58     }
59 }
60
```



Als laatste moeten we het script nog toe gaan voegen aan het project om alles te laten werken. Dit script werkt vanaf een object wat al in de Scene staat. Zorg ervoor dat het object dat je wil laten zien in de Scene staat en ga naar de inspector van dit object.



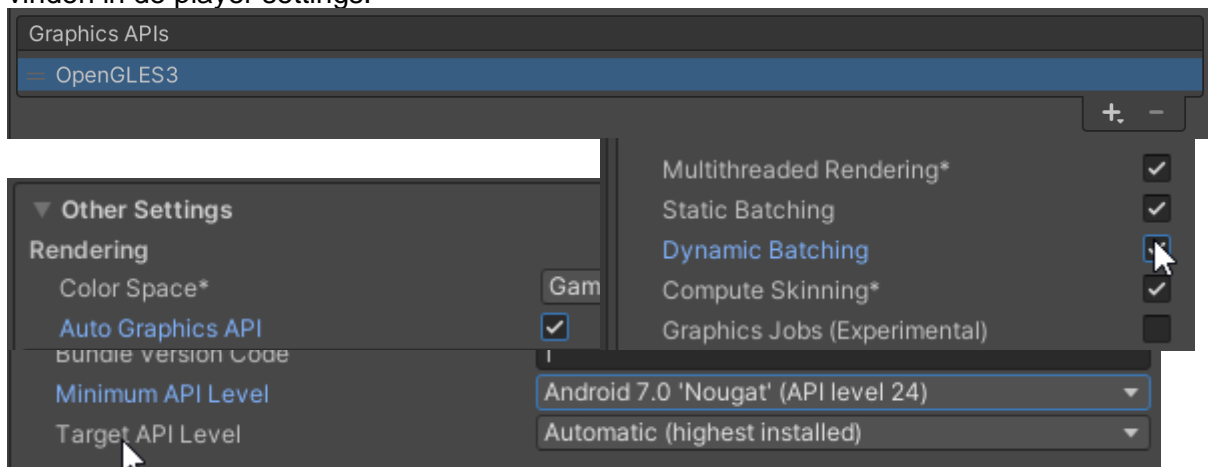
In het project folder ga je opzoek naar het script wat zojuist is aangemaakt. Sleep het script in de inspector van het object.



We hebben hier een enkele input. Gebruik dit om de naam uit de ReferenceImageLibrary in te typen. Let op dat de naam exact hetzelfde moet staan zoals het in de ReferenceImageLibrary staat.

PLAYER SETTINGS TROUBLESHOOTING;

Zijn er problemen met het bouwen? Kijk of de volgende settings correct staan. Deze zijn te vinden in de player settings.



BEOORDELING

De module wordt beoordeeld op de leerdoelen. Per opdracht staat (bij de opdracht) aangegeven om welk(e) leerdoel(en) het gaat. Hoeveel punt je per leerdoel kan behalen zie je in het beoordelingsformulier hieronder. Op basis van de leerdoelen waarvoor je punten hebt ontvangen worden ook werkprocessen afgetekend.

LEERDOELEN

De leerdoelen zijn:

Je kunt:

- Een goede planning opstellen en gebruiken
- In een team samenwerken / overleggen
- Een product presenteren
-
-
-

WERKPROCESSEN

De werkprocessen zijn:

- B1-K1-W1 : Plant werkzaamheden en bewaakt de voortgang
- B1-K1-W3 : Realiseert (onderdelen van) software
- B1-K1-W4 : Test software
- B1-K2-W1 : Voert overleg
- B1-K2-W2 : Presenteert het opgeleverde werk

De werkprocessen worden individueel beoordeeld.

Op de volgende pagina staat het gebruikte beoordelingsformulier afgedrukt.

BEOORDELINGSFORMULIER

Module		
Naam beoordelaar		
Naam student		
Datum		
Opdracht:	Leerdoelen:	Punten:
	<ul style="list-style-type: none"> Een goede planning opstellen en gebruiken In een team samenwerken / overleggen Een product presenteren 	(max 30)
1 kassa-systeem	<ul style="list-style-type: none"> 	(max 15)
2 bestelzuil	<ul style="list-style-type: none"> 	(max 15)
3 mensen tellen – product inventaris	<ul style="list-style-type: none"> 	(max 15)
4 hamburger op tafel maken	<ul style="list-style-type: none"> 	(max 15)
	Extra onderdelen	(max 10)

Opmerkingen:	
Bij te laat inleveren	Maximum cijfer: 6
Niets inleveren	Cijfer: 1
Cijfer:	

Werkproces	O/V
B1-K1-W1 : Plant werkzaamheden en bewaakt de voortgang	
B1-K1-W3 : Realiseert (onderdelen van) software	
B1-K1-W4 : Test software	
B1-K2-W1 : Voert overleg	
B1-K2-W2 : Presenteert het opgeleverde werk	
Feedback:	