# Лек Техніка розпізнавання обличь в AR

Викладач Ph.D, в.о. зав.каф. КН Граф М.С.

План

- 1. Відстеження облич в AR
- 1.1. Вступ
- 1.2. Налаштування проекту
- 1.3. Додавання обличчя AR за замовчуванням
- 1.4. Імпортування текстур для використання AR Faces користувача
- 1.5. Створення додаткових облич
- 2. Генератор випадкових зображень з AR Face Tracking
- 2.1. Пошук зображення
- 2.2. Кодування генератора зображень
- 2.3. Завершення налаштувань кодування зображень

### 1.1. Вступ

- Використовуйте AR Face Tracking, щоб створити накладення обличчя AR за замовчуванням
- Додайте матеріали на поверхню, щоб створити власне обличчя



### 1.1. Вступ



# 1.2.Налаштуванняпроекту

Package Manager			
+ ▼ All packages ▼	Advar	nced	•
Adaptive Performance Samsun	1.1.9		•
Addressables	1.8.5		
Advertisement	3.4.9		
Alembic	1.0.7		
Analytics Library	3.3.5		
Android Logcat	1.1.1		
AR Foundation	2.1.10		
	4.1.1		
See all	versio	ns	
AR Subsystems	2.1.3	$\checkmark$	
ARCore XR Plugin	2.1.12		
ARKit Face Tracking	1.0.8		
ARKit XR Plugin	2.1.10		
Asset Bundle Browser	1.7.0		
Barracuda	1.0.4		
► Burst	1.2.3		
Cinemachine	2.7.1		
► Core RP Library	7.3.1		
Custom NUnit	1.0.0	J	
Editor Coroutines	1.0.0		-
Last update Nov 22, 14:21		C	2

### **AR Foundation**

Version 4.1.1

### Name

com.unity.xr.arfoundation

### Links

View documentation

/iew licenses

### Author

Unity Technologies Inc.

### **Published Date**

November 17, 2020

A collection of MonoBehaviours and C# utilities for working with AR Subsystems.

#### Includes:

- GameObject menu items for creating an AR setup
- MonoBehaviours that control AR session lifecycle and create GameObjects from detected, real-world trackable features
- Scale handling
- Face tracking

Samples available at



# 1.2.Налаштуванняпроекту

Package Manager			
+ ▼ All packages ▼	Advar	nced	•
2D Path	2.0.6		^
2D Pixel Perfect	2.0.4		
2D PSD Importer	2.1.6		
2D Sprite	1.0.0		
2D SpriteShape	3.0.14		
2D Tilemap Editor	1.0.0		
Adaptive Performance	1.1.9		
Adaptive Performance Samsun	1.1.9		
Addressables	1.8.5		
Advertisement	3.4.9		
Alembic	1.0.7		
Analytics Library	3.3.5		
Android Logcat	1.1.1		
AR Foundation	2.1.10	$\checkmark$	
AR Subsystems	2.1.3	$\checkmark$	
ARCore XR Plugin	2.1.12		
ARKit Face Tracking	1.0.8		
ARKit XR Plugin	2.1.10	$\checkmark$	
Asset Bundle Browser	1.7.0		
► Barracuda	1.0.4		
D.u.st	100		•
Last update Nov 19, 10:32		C	



Remove

### **ARKit Face Tracking**

Version 1.0.8 (2019.4 verified)

### Name

com.unity.xr.arkit-face-tracking

### Links

View documentation

### iew licenses

### Author

Unity Technologies Inc.

### **Published Date**

October 20, 2020

Support for face tracking on ARKit.

Includes support for:

- Face pose tracking
- Blendshapes

Note: Only supported on devices with a forward-facing depth camera, like the iPhone X

### 1.2. Налаштування проекту

2.1. Створіть нову сцену у своєму проекті, клацнувши правою кнопкою миші на вікні проекту: Створити > Сцена.

2.2. Видаліть основну камеру, клацнувши правою кнопкою миші та вибравши Видалити. Клацніть правою кнопкою миші в ієрархії, щоб створити новий AR Session XR > AR Session. Потім додайте джерело сеансу AR, клацнувши правою кнопкою миші в Ієрархії XR > Початок сеансу AR. Тепер ієрархія має вигляд:



2.3. Після цього виберіть камеру AR в ієрархії (ви можете знайти її, натиснувши стрілку поруч із **джерелом** сеансу AR ) і в інспекторі встановіть тег як основну камеру.

## 1.3. Додавання обличчя AR за замовчуванням

Inspector				6	:
🕥 🗹 🗛 Sessio	on Origin		s	Statio	с 🔻
Tag Untagged	<b>•</b>	Layer Default			▼
<b>人</b> Transform			9	ᅷ	:
Position	X 0	Y O	ZO		
Rotation	X 0	Y O	ZO		
Scale	X 1	Y 1	Z 1		
🔻 💀 🔽 AR Session	Origin (Script)		0	÷	:
Script	ARSess	ionOrigin			$\odot$
Camera	AR Cam	nera (Camera)			$\odot$
	Add Compo	nent			
् AR fa	ce	×	1		
	Search				
🖸 AR F	ace				
🥺 AR F	ace Manager				
🥺 AR F	ace Mesh Visua	lizer			
New	script	>	•		

# 1.3. Додавання обличч AR за замовчуванням



## 1.3. Додавання обличчя AR за замовчуванням

🖿 Project	E Console		
+-			
Folder		ssets	
C# Script Shader Testing Playables Assembly Assembly TextMeshF	Definition Definition Reference Pro	Scenes	XR
Scene Prefab Var	iant		
Audio Mixe	er		
Material Lens Flare Render Te: Lightmap I Custom Re	xture Parameters ender Texture		

Sprite Atlas



## 1.3. Додавання обличч AR за замовчуванням

Inspector			6	:
AR Session Ori	gin Layer Default		Stati	c ▼
🔻 🙏 🛛 Transform		8	ᅷ	:
Position	X 0 Y 0	ZO		
Rotation	X 0 Y 0	Z 0		
Scale	X 1 Y 1	Z 1		
🔻 🥺 🖌 AR Session Orig	in (Script)	8	÷	:
Script	ARSessionOrigin			$\odot$
Camera	AR Camera (Camera)			$\odot$
🔻 🥺 🖌 AR Face Manage	er (Script)	8	÷	:
Script	ARFaceManager			۲
Face Prefab	🕆 AR Default Face			$\odot$
Maximum Face Count	1			

# 1.3. Додавання обличч AR за замовчуванням

Inspector			Ъ	:
AR Camera		) 🗌 s	tati	с▼
Tag MainCamera	<ul> <li>Layer Default</li> </ul>			•
J Transform		9	ᅷ	:
🕨 💶 🔽 Camera		9	ᅷ	:
🕨 🥺 🔽 AR Pose Driver (	Script)	9	ᅷ	:
🔻 🥺 🔽 🗛 Camera Man	ager (Script)	8	ᅷ	:
Script	ARCameraManager			$\odot$
Auto Focus	✓			
Light Estimation	None			▼
Facing Direction	User			▼

# 1.3. Додавання обличч: AR за замовчуванням

		Build Settings		
Scer	nes In Build			
✓ So So	enes/AR Face enes/SampleScene			0
				Add Open Scenes
Platf	form	7		
	PC, Mac & Linux Standalone	iOS ios		
iOS	S ios 🔫	Run in Xcode	Latest v	ersion 🔻
E	WebGL	Run in Xcode as Symlink Unity librarie	Release	•
tvOS	S tvOS	Development Build Autoconnect Profiler		
∠רק	1 PS4	Deep Profiling Script Debugging		
Ŵ	Xbox One	Scripts Only Build		
	Android	Compression Metho	d Default	<b>•</b>
			Learn abo	out Unity Cloud Build
Play	yer Settings		Build	Build And Run

### 1.3. Додавання обличчя AR за замовчуванням

Якщо у вас виникли проблеми з тестуванням на своєму пристрої Android, перегляньте <u>документацію Google,</u> щоб оновити версію Gradle.



# 1.4. Імпортування текстур для використання AR Faces користувача

Home > Essentials > Asset Packs > AR Face Assets







## 1.4. Імпортування текстур для використання AR Faces користувача

# 1.4. Імпортування текстур для використання AR Faces користувача



# 1.4. Імпортування текстур для використання AR Faces користувача

Inspector			3	:
AR Session Ori	igin	] 🗌 s	Stati	с▼
Tag Untagged	<ul> <li>Layer Default</li> </ul>			•
J Transform		9	÷	:
🔻 🥺 🖌 AR Session Orig	in (Script)	•	井	:
Script	ARSessionOrigin			۲
Camera	AR Camera (Camera)			$\odot$
🔻 🥺 🖌 AR Face Manage	er (Script)	9	÷	:
Script	ARFaceManager			$\odot$
Face Prefab	📦 Pop Face			$\odot$
Maximum Face Count	1			



1.4. Імпортування текстур для використання AR Faces користувача

# 1.4. Імпортуваннятекстур длявикористання ARFaces користувача

0	Inspector				6
6	💧 Pop Fa	ce			0 ‡ ≎
	Shader	Standard			▼
	Renderina M	ode	Opaque		•
	Main Maps				
	⊙ Albedo		58		
	⊙Metallio	C	•		0
	Smooth	ness	<b>●</b>		0.5
	Sour	ce	Metallic Alpha		•
	☉Normal	Мар			
	☉ Height	Мар			
	⊙Occlusi	ion			
	⊙Detail N	Mask			
	Emission				
	Tiling		X 1	Y 1	
	Offset		X 0	Y -0.045	



1.4. Імпортування текстур для використання AR Faces користувача

# 1.5. Створення додаткових облич

Inspector			9	:
Robot Face		S	tati	с 🕶
Tag Untagged	▼ Layer Default			•
J Transform		8	÷	:
🕨 🥺 🗹 AR Face (Script)		8	ᅷ	:
🕨 🥺 🔽 AR Face Mesh V	isualizer (Script)	8	ᅷ	:
🕨 🌐 🗹 Mesh Collider		•	<u>-</u> +-	:
▶ 🌐 [none] (Mesh Filt	ter)	9	井	:
🔻 🖽 🗹 Mesh Renderer		•	÷	:
Materials				
Size	1			
Element 0	Robot_Albedo			$\odot$
Lighting				
Cast Shadows	Off			▼
Receive Shadows				
Contribute Global Illui				
Receive Global Illumiı	Light Probes			▼
Probes				
Light Probes	Blend Probes			▼
Reflection Probes	Blend Probes			▼
Anchor Override	None (Transform)			$\odot$
Additional Settings				
Motion Vectors	Per Object Motion			▼
Dynamic Occlusion				
Robot_Albedo			97	± 🌣
Shader Standard	d			•

# 1.5. Створення додаткових

облич

Project	E Console		
+•			
🔻 🛨 Favorite	es	Assets > Mate	erials
🔍 🔍 All Ma	aterials		
🔍 All Mo	odels		
Q All Pre	efabs		
Assets			
🖿 Mater	tials	Pop Face	Robot_Alb
🖿 Prefa	bs		
🖿 Scene	es		
🖿 Textu	res		
🕨 🖿 XR			
▶ 🖿 Packag	es		



# 1.5. Створення додаткових облич

## 1.5. Створення додаткових облич



### 1.5. Створення додаткових

облич

Inspector			a :
Robot_	Albedo		0 ‡ ¢
Shader	Standard		•
Rendering M <b>Main Maps</b>	ode	Opaque	•
⊙Albedo		54	
⊙ Metallio	c l	•	0
Smooth	nness	•	0.5
Sour	ce	Metallic Alpha	•
💽 $\odot$ Normal	Мар		1
○ Height	Мар		
😻 🛛 Occlusi	ion	•	1
🗌 🛛 Detail N	Mask		

### 2. Генератор випадкових зображень з AR Face Tracking

**AR Face Assets** 

## 2. Генератор випадкових зображень з AR Face Tracking

### 2.1. Пошук зображення



Роздільна здатність рекомендуєтьс 320\*180 пікселів



0	Inspector			6
	createw	vithcode Ir	nport Settings	❷ ∓± ✿ Open
Т	Texture Type		Sprite (2D and UI)	<b>~</b>
Т	exture Shape		2D	<b>*</b>
S	prite Mode		Single	•
	Pixels Per Unit		1000	
	Mesh Type Extrude Edges		Tight	•
			•	1
	Pivot		Center	•
	Generate Physics Sh			
				Sprite Editor
► A	Advanced			Sprite Editor
► A V	Advanced Vrap Mode		Clamp	Sprite Editor
► A V F	Advanced Vrap Mode Filter Mode		Clamp Bilinear	Sprite Editor
► A V F A	Advanced Vrap Mode Filter Mode Aniso Level		Clamp Bilinear	Sprite Editor
► A V F A	Advanced Vrap Mode Filter Mode Aniso Level Default	Ē	Clamp Bilinear	Sprite Editor
► A V F A	Advanced Vrap Mode Filter Mode Aniso Level Default Max Size		Clamp Bilinear iOS 2048	Sprite Editor
► A V F	Advanced Vrap Mode Filter Mode Aniso Level Default Max Size Resize Algor		Clamp Bilinear iOS 2048 Mitchell	Sprite Editor
► A V F	Advanced Vrap Mode Filter Mode Aniso Level Default Max Size Resize Algor Format	ithm	Clamp Bilinear iOS 2048 Mitchell Automatic	Sprite Editor
► A V F	Advanced Vrap Mode Filter Mode Aniso Level Default Max Size Resize Algori Format Compressior	ithm	Clamp Bilinear iOS 2048 Mitchell Automatic Normal Quality	Sprite Editor

Revert Apply



<sup>™</sup> ⊟ Hierarchy			a :			
+▼ Qr All						
🔻 🍕 AR Fac	e*		:			
🕥 Direc	ctional Light					
💮 AR S	Session					
🕨 💮 AR S	ession Origin					
🔻 💮 Imag	je Randomizer					
🕥 lm	age					
		Inspector				a :
	6	🕤_ 🗹 Imaç	je			Static 🔻
		Tag Unta	agged	▼ La	yer Default	•
		📕 🕹 🕹	form			0 7 i
		Position	Х	0	Y 0.25	Z 0
		Rotation	Х	0	Y 0	Ζ 0
		Scale	X	1	Y 1	Z 1

### 2.2. Кодування генератора зображень



### 2.2. Кодування генератора зображень

public SpriteRenderer randomSpriteRenderer;

//Рендерер спрайтів, який відображатиме випадкове зображення

public Sprite[] randomSprites;

//Масив спрайтів, що будуть використовуватися для рандомізації зображення

public float timeBetweenChange = 0.2f; //Кількість часу між перемиканням зображення public float timeUntilStopping = 3.0f;

//Загальна кількість часу до припинення рандомізації

private int RandomImageIndex = 0; //Слідкуйте за тим, яке зображення зараз показується private float ImageChangeTimer; //Скільки часу залишилося до переходу на нове зображення

void Start()

// Set this timer variable as soon as the script starts so it's ready for the update function
ImageChangeTimer = timeBetweenChange;

### 2.2. Кодування генератора зображень

### void Update()

```
// Remove delta time from both active timers. This subtracts a small amount of time from the
// overall time we described in "timeBetweenChange" and "timeUntilStopping«
     ImageChangeTimer -= Time.deltaTime;
     timeUntilStopping -= Time.deltaTime;
// If we've fully run out of time we need to select a final random image
if(timeUntilStopping <= 0.0f)
// The final image will be selected at random
             randomSpriteRenderer.sprite = randomSprites[Random.Range(0, randomSprites.Length)];
// Destroy this script immediately stopping it from running anymore in the future
             DestroyImmediate(this);
// Return so no other code runs in this Update function call
return;
// If RandomizationTimer is less than or equal to zero it's time for a new image
if(ImageChangeTimer <= 0.0f)</pre>
// To ensure we see all the images we increase the RandomImageIndex to see the next image in the array
             RandomImageIndex++;
// If our index has gone past the end of the array, reset it to zero so the cycle can start again
if(RandomImageIndex >= randomSprites.Length)
                RandomImageIndex = 0;
//Assign the new sprite to the sprite renderer
           randomSpriteRenderer.sprite = randomSprites[RandomImageIndex];
//Reset the "RandomizationTimer" to start counting down again
           ImageChangeTimer = timeBetweenChange;
```

### 2.3. Завершення налаштувань кодування зображень

Inspector					а	:	
🕥 🗹 In	Image Randomizer				Static 🔻		
Tag U	Intagged	▼ La	ayer Default			•	
🔻 🙏 🔹 Tra	Insform			•	- <b>t</b> -	:	
Position	Х	0	Y 0	ZO			
Rotation	Х	0	Y 0	Ζ0			
Scale	Х	1	Y 1	Z 1			
	Add Component			]			
	Search						
	# Image Randomizer						
	New script		>				

2.3. Завершенняналаштуванькодуваннязображень

Inspector	Ċ	3:		
🕎_ 🗹 Image Random	izer		Sta	tic 🔻
Tag Untagged	▼ La	ayer Default		•
🔻 🙏 🛛 Transform			? ∓	± :
Position	X 0	YO	Z 0	
Rotation	X 0	Y 0	ZO	
Scale	X 1	Y 1	Z 1	
🔻 📕 🔽 Image Randomizer (Script)			€ 7	± :
Script	#ImageRan	domizer		$\odot$
Random Sprite Renderer	ទៅ Image (Sp	rite Renderer)		$\odot$
Random Sprites				
Size	3			
Element 0	Element 0 II createwithcode			
Element 1	⊡johnlemon			
Element 2	⊡ rollaball			$\odot$
Time Between Change	0.2			
Time Until Stopping 3				

### 2.3. Завершення налаштувань кодування зображень



# Коротка інструкція по створенню масок доповненої реальності в Unity

Інструкція по створенню масок доповненої реальності в Unity:

Крок 1: Підготовка до роботи

Перш за все, вам потрібно завантажити та встановити останню версію Unity з офіційного сайту. Ви можете завантажити безкоштовну версію Unity, що відповідає вашим потребам. Крім того, вам потрібен плагін AR, такий як ARCore або Vuforia, який дозволить вам працювати з AR-функціями.

Крок 2: Створення нового проекту

Після встановлення Unity запустіть програму і створіть новий проект. У вікні "Новий проект" виберіть "3D" або "2D" залежно від того, який тип AR-додатка ви хочете створити.

Крок 3: Додавання AR-об'єктів

Для створення AR-додатка вам потрібно додати AR-об'єкти в проект. Щоб це зробити, ви можете використовувати плагіни ARCore або Vuforia. Обидва плагіни дозволяють додавати AR-об'єкти в проект та настроювати їх. Крок 4: Створення масок

Щоб створити маски доповненої реальності в Unity, вам потрібно використовувати спеціальні шейдери, які дозволяють відображати текстуру на об'єктах в режимі реального часу. Шейдери дозволяють вам налаштовувати свої маски, додаючи різні фільтри та ефекти.

Крок 5: Додавання взаємодії з масками

Після створення масок ви можете додати взаємодію з ними, таку як переміщення, обертання та зміну розміру. Ви можете налаштувати взаємодію з масками за допомогою скриптів, що дозволить вам створити реалістичну інтерактивність.

## Коротка інструкція по створенню масок доповненої реальності в Unity

6.Відкрийте вікно "Inspector" для об'єкту "AR Camera", встановіть прапорець "Vuforia Augmented Reality Supported". 7.Відкрийте вікно "Vuforia Configuration" через меню "Window" -> "Vuforia Configuration".

8.Введіть свій ключ API в розділі "App License Key".

9.Вкажіть, яку маркерну систему ви хочете використовувати в розділі "Database" (на цьому етапі ви можете додати нову маркерну систему або використовувати вже наявну).

10.3мініть настройки для камери в розділі "Camera Device".

11.Відкрийте вікно "Project" та створіть новий об'єкт (наприклад, "Cube") для відображення маски.

12.Відкрийте вікно "Inspector" для об'єкту "Cube" та настройте його відображення за вашим бажанням.

13.Відкрийте вікно "Hierarchy" та перетягніть об'єкт "Cube" під об'єкт "AR Camera".

14.Відкрийте вікно "Inspector" для об'єкту "Cube" та встановіть прапорець "Image Target Behaviour" в розділі "Add Component".

15.Виберіть маркерну систему, яку ви використовуєте, та виберіть потрібний маркер в розділі "Image Target Behaviour".

16.Відкрийте вікно "Scene" та запустіть симуляцію (натисніть кнопку "Play" внизу середини екрану).

17. Наведіть камеру на маркер та перевірте, чи відображається маска на об'єкті "Cube".

Це загальна інструкція, яка допоможе вам створити маски доповненої реальності в Unity з використанням Vuforia. Залежно від ваших потреб, вам можуть знадобитися додаткові кроки або інші налаштування. Рекомендується ознайомитися з документацією Unity та Vuforia для отримання додаткової інформації.

# Кроки для створення масок доповненої реальності в Unity на обличчя, що можуть рухатися

1. Створіть новий проект в Unity та додайте залежності AR Foundation та ARKit або ARCore (залежно від платформи, на якій ви плануєте запускати додаток).

2. Додайте камеру AR к сцені. Ця камера служитиме для відстеження положення і орієнтації обличчя в режимі реального часу.

3.Додайте компонент AR Face Manager до камери AR. Цей компонент дозволяє відстежувати рухи обличчя в режимі реального часу.

4. Додайте модель маски до сцени. Ви можете створити власну модель маски або завантажити її з веб-сайту, що пропонує безкоштовні моделі.

5.Додайте текстуру до моделі маски. Текстура повинна мати прозорий фон, щоб дозволити показувати обличчя знизу. 6.Створіть новий матеріал для моделі маски та додайте до нього текстуру.

7. Додайте компонент AR Face Attach до моделі маски. Цей компонент дозволяє моделі прив'язуватися до обличчя. 8. Використовуйте скрипт для визначення положення та орієнтації моделі маски на обличчі. Цей скрипт можна знайти в AR Foundation Samples або ARKit / ARCore Samples.

9. Додайте звуковий ефект до маски, якщо потрібно.

10.3бережіть проект та експортуйте його для використання на платформі, на яку ви спрямовуєтеся.

Зверніть увагу, що це загальні кроки, і деталі можуть змінюватися залежно від платформи та конкретного проекту. Інструкція має на меті надати загальний огляд процесу створення масок доповненої реальності в Unity на обличчя.

### Інструкція по створенню масок доповненої реальності в Unity на декілька облич одразу

Для створення маски доповненої реальності на декілька облич одразу, можна використати бібліотеку ARCore от Google. Ось інструкція, яка допоможе створити додаток:

1. Створіть новий проект в Unity та додайте пакет ARCore. Для цього відкрийте меню "Window" > "Package Manager" та встановіть пакет ARCore.

2.Додайте ARCore Face Prefab до сцени. Для цього перетягніть префаб "ARCore Face" з папки

"Assets/GoogleARCore/Examples/FaceRegions/Prefabs" до сцени.

3.3мініть налаштування ARCore Face Prefab. Виберіть ARCore Face Prefab на сцені та знайдіть компонент "AugmentedFaceVisualizer" у його дочірніх об'єктах. Встановіть параметр "Max Number of Faces" на потрібне значення (наприклад, 2 для двох облич).

4. Створіть маску для кожного облича. Створіть новий матеріал та додайте до нього текстуру для маски. Використовуйте "Texture Type" "Advanced" та встановіть параметр "Alpha Source" на "Input Texture Alpha". Додайте цей матеріал до префабу "FaceMeshVisualizer".

5.Налаштуйте показ маски на кожному обличі. Виберіть кожен дочірній об'єкт "AugmentedFaceMeshVisualizer" та знайдіть компонент "MeshRenderer". Встановіть для нього матеріал, створений в кроці 4.

6.Запустіть додаток та спробуйте рухати обличчя перед камерою. Маски повинні з'являтися на кожному обличі. Дана інструкція допоможе вам створити маску доповненої реальності на декілька облич одразу в Unity з використанням ARCore.

### Дякую за увагу!