



สร้างเกม 2 มิติด้วย Unity

(สำหรับผู้เริ่มต้น)

รู้จักกับโปรแกรม Unity



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

Unity คืออะไร

Unity คือ Game Engine หรือ เครื่องมือสำหรับการพัฒนาเกมได้ทั้งรูปแบบ 2 มิติและ 3 มิติที่สามารถทำงานข้าม Platform ได้ ทั้งแบบ Desktop (Windows, Mac, Linux) Mobile (iOS, Android) และ Web (HTML5 | WebGL)

Unity คืออะไร

ผู้ใช้สามารถสร้างเกมให้รองรับใน Platform ดังกล่าว
ได้โดยการพัฒนาเกมเพียงครั้งเดียว ตัว Engine
ประกอบด้วย API ในการช่วยเหลือนักพัฒนาด้านการเขียน
Script โปรแกรม โดยภาษาโปรแกรมที่ใช้เขียนคือ C#

Unity คืออะไร

ในปัจจุบัน Unity ได้ถูกนำไปใช้ในงานด้านต่างๆที่ไม่ได้
เกี่ยวข้องกับการทำเกมอย่างเดียว เช่น งานด้านภาพยนตร์
, สถาปัตยกรรม , วิศวกรรม รวมไปถึงการโต้ตอบกับผู้ใช้
โดยการพัฒนาในรูปแบบของงานด้าน AR และ VR

Unity คืออะไร

การใช้งานโปรแกรม Unity สามารถใช้งานได้ฟรี ไม่เสียค่าใช้จ่าย แต่อาจจะมีบาง Feature ที่ต้องทำการชำระเงินถึงจะใช้งานได้ เช่น Package เสริมบางตัว โดยต้องทำการสมัครสมาชิกเพื่อรับใบอนุญาตฟรีสำหรับงานส่วนบุคคลหรือบริษัทขนาดเล็ก



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

ต้องมีพื้นฐานอะไรบ้าง ?

ต้องมีพื้นฐานอะไรบ้าง

- การเขียนโปรแกรมภาษา C#
 - ไวยากรณ์พื้นฐาน
 - การเขียนโปรแกรมเชิงวัตถุ

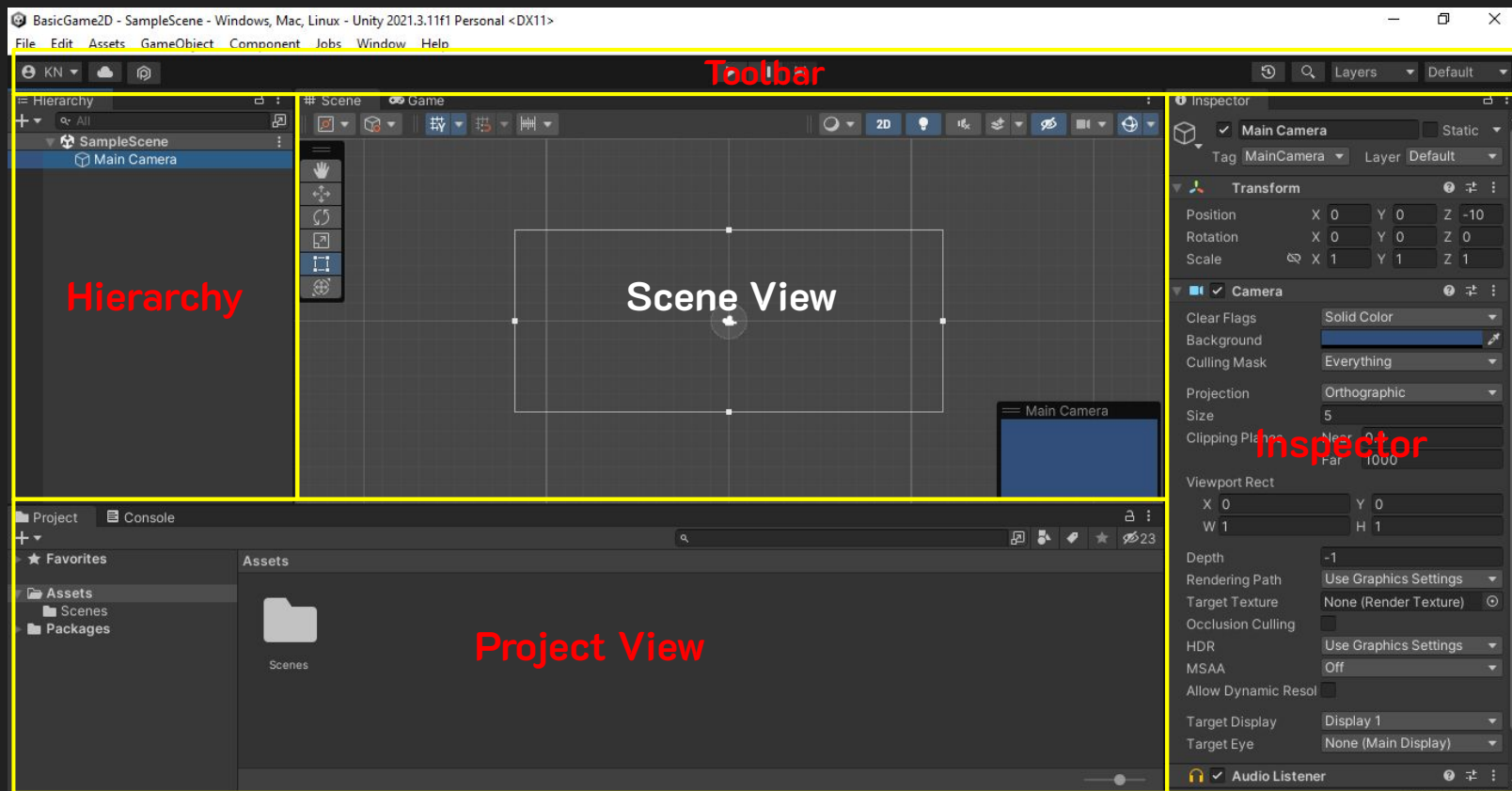


เครื่องมือที่ใช้

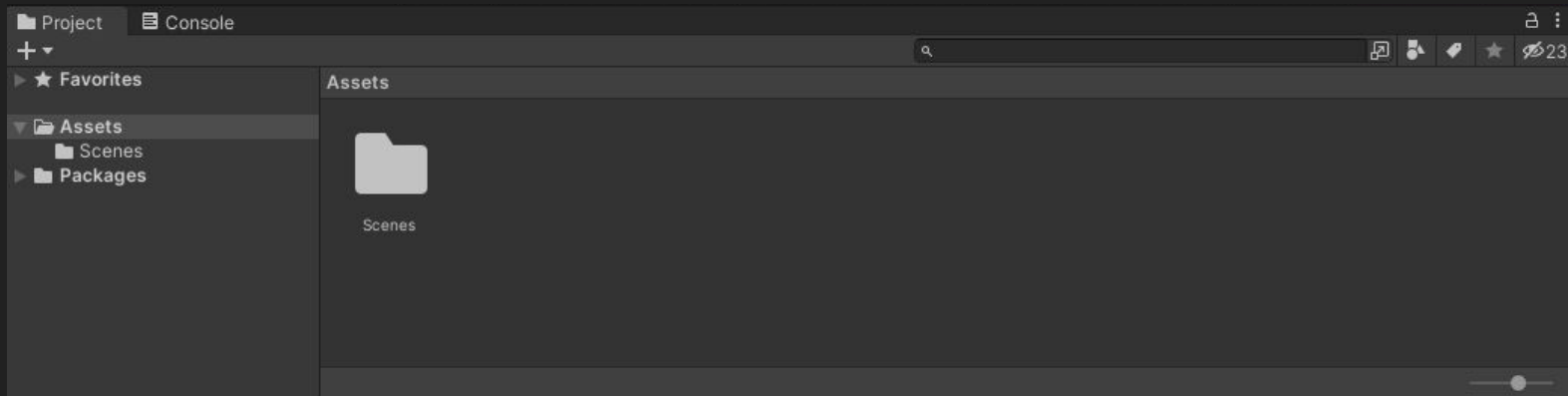
- Unity 2021 เป็นต้นไป
- Visual Studio Community 2019

องค์ประกอบของโปรแกรม Unity

หน้าจอของโปรแกรม

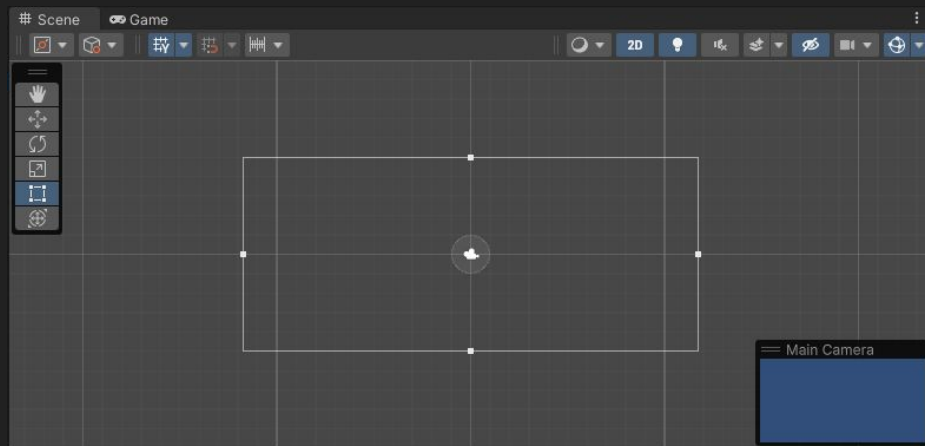


Project View



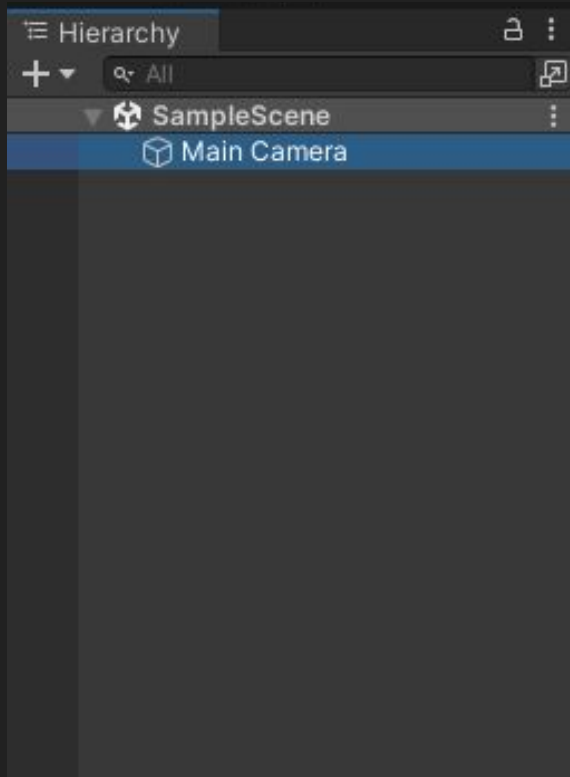
เป็นหน้าต่างที่แสดงโฟลเดอร์และไฟล์ต่างๆที่เก็บและใช้งานในโปรเจกต์ เช่น โมเดล , ภาพ , เสียง , Scene รวมไปถึง Script ไฟล์ จะเรียกส่วนนี้ว่า Asset หากต้องการนำไฟล์จากด้านนอกมาทำงานต้องลากมาไว้ที่โปรเจกต์

Scene View



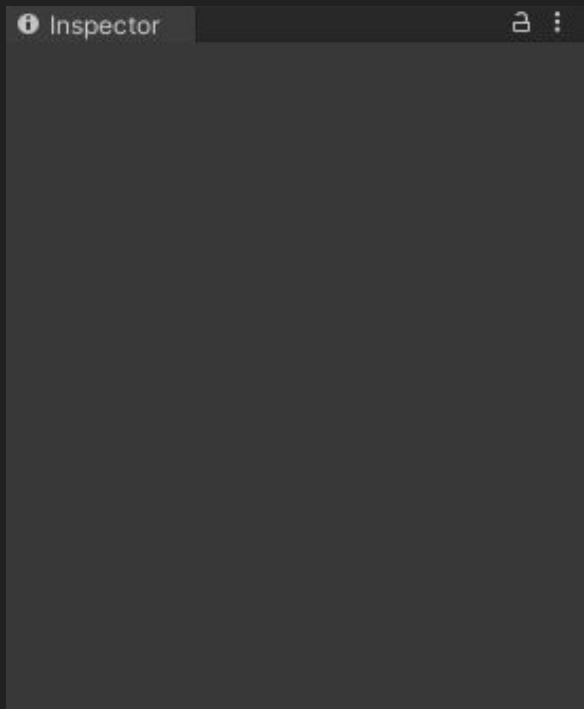
ส่วนที่ใช้จัดมุมมองในการออกแบบเกมขึ้นอยู่กับว่าจะแสดงมุมมอง 2 มิติ หรือ 3 มิติ รวมไปถึงการปรับแต่งวัตถุใน Scene เช่น การหมุน ย่อ- ขยายวัตถุ นักพัฒนาสามารถที่จะลาก Asset เข้าไปใน Scene View ได้ซึ่งส่วนที่ออกแบบใน Scene จะถูกแสดงในส่วนหน้าจอเกมด้วย

Hierarchy



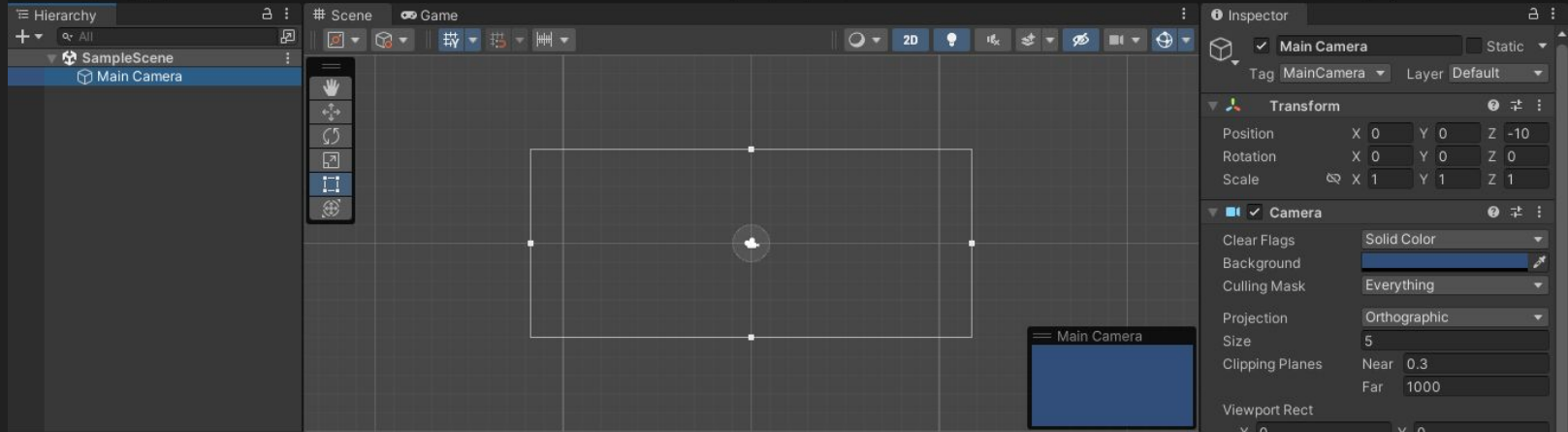
ส่วนที่แสดงลำดับชั้นของวัตถุที่ปรากฏใน Scene View โดยจะบอกชื่อวัตถุที่ทำงานภายใน Scene View ว่ามีวัตถุอะไรบ้าง เช่น Main Camera (กล้อง) เป็นต้น

Inspector



หน้าต่างอำนวยความสะดวกในการจัดการของคุณสมบัติทั้งหมดของวัตถุที่เรา กำลังทำงานด้วย เช่น การกำหนดตำแหน่ง หมุนย่อ - ขยาย วัตถุ การตั้งชื่อวัตถุ การใส่ Tag หรือการเพิ่ม Component ลงไปใน วัตถุ เช่น ไฟล์ Script เป็นต้น

Inspector



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

Toolbar



แถบเครื่องมือพื้นฐานสำหรับกำหนดคุณสมบัติต่างๆที่อยู่ภายใน Scene
เกม เช่น การปรับตำแหน่ง , ย่อ-ขยาย , หมุน วัตถุการควบคุมการเล่นหรือ
หยุดเกมและการจัดการบัญชีผู้ใช้

คำศัพท์พื้นฐาน

คำศัพท์พื้นฐาน

Asset หมายถึงสิ่งที่นำเข้ามาทำงานใน Project สำหรับพัฒนาเกม ยกตัวอย่าง เช่น Model ตัวละคร, Animation Script, Texture , เสียงและอื่นๆ ซึ่ง Asset จะถูกเก็บไว้ใน Project มีชื่อไฟล์เดอร์ว่า Assets ถ้าหากไม่ได้อยู่ในโฟลเดอร์นี้ก็ไม่สามารถพัฒนาเกมได้



คำศัพท์พื้นฐาน

Scene (ฉาก) หมายถึง หน้าจอแสดงผลหรือฉากการทำงานของเกมโดยภายใน 1 เกมสามารถแบ่งฉากออกเป็นหลายๆฉากได้ เช่น ฉากเมนู , ฉากเปิดเกม , ฉากเนื้อเรื่อง ฉากจบเกม เป็นต้น



คำศัพท์พื้นฐาน

Camera หมายถึง กล้องที่ใช้จับภาพต่างๆภายในเกม ภาพที่แสดงผลมีมุมมองอย่างไรขึ้นอยู่กับนักพัฒนาเกม โดยภายในเกมสามารถมีกล้องได้มากกว่า 1 ตัว ถ้าไม่มีกล้องก็จะมีภาพปรากฏในเกมนั่นเอง



คำศัพท์พื้นฐาน

Light หมายถึง วัตถุประเภทแสง ใช้ปรับความมืด
ความสว่างภายในเกมรวมไปถึงเงาของวัตถุ

คำศัพท์พื้นฐาน

GameObject หมายถึง วัตถุหรือองค์ประกอบต่างๆที่อยู่ภายในเกม เช่น คน สัตว์ สิ่งของ เป็นต้น

- **Empty Object** วัตถุว่างเปล่าที่ไม่ปรากฏใน Scene เกมนิยมนำมาใช้สำหรับรัน Script ไฟล์ , จัดกลุ่มวัตถุเรียกใช้งาน Component ต่างๆเช่น การเล่น Sound Background , การหยุดเกม (Play/Pause) เป็นต้น



คำศัพท์พื้นฐาน

Component หมายถึง คุณสมบัติต่างๆที่อยู่ใน GameObject เช่น ระบบฟิสิกส์ ระบบควบคุม Animation , ระบบเสียง หรือ Script ต่างๆที่อยากให้ GameObject นั้นสามารถทำงานตามที่ต้องการ

คำศัพท์พื้นฐาน

Transform หมายถึง Component ที่ต้องมีอยู่ในวัตถุทุกตัวเป็นสิ่งที่ขาดไม่ได้ โดยจะเป็นตัวที่เก็บค่า 3 ค่าได้แก่

- **Position** คือตำแหน่งของวัตถุ (พิกัด x, y, z ถ้าอยู่ในรูป 3 มิติ)
- **Rotation** คือการหมุนวัตถุ (พิกัด x, y, z ถ้าอยู่ในรูป 3 มิติ)
- **Scale** คือขนาดของวัตถุ (พิกัด x, y, z ถ้าอยู่ในรูป 3 มิติ)

คำศัพท์พื้นฐาน

Rigidbody / Rigidbody2D หมายถึง Component ที่จัดการด้านระบบฟิสิกส์ในวัตถุ เช่น มวล(mass) , แรงโน้มถ่วง (Gravity) , การเคลื่อนไหวของวัตถุ (Kinematic) , การล็อควัตถุ (Freeze)

Sprite คือ ภาพที่ทำมาใช้งานในระบบเกม (2 มิติ)



คำศัพท์พื้นฐาน

Particle System หมายถึง ระบบ Effect ภายในเกม เช่น Effect ระเบิด , Effect ไฟ , ฝุ่น ใช้เพิ่มสีสันภายในเกม

Texture หมายถึง รูปภาพที่นำมาประกอบในโมเดลให้มีความสมจริงมากยิ่งขึ้น

Material หมายถึง เม็ดสีที่ปรากฏในตัววัตถุ



คำศัพท์พื้นฐาน

Vector2 คือ ตัวแปรที่เก็บตำแหน่งแกน X , Y ในระบบ
เกมแบบ 2 มิติ

Vector3 คือ ตัวแปรที่เก็บตำแหน่งแกน X , Y , Z ใน
ระบบเกมแบบ 3 มิติ

คำศัพท์พื้นฐาน

Collider / Collider2D หมายถึง Component ที่ใช้ตรวจสอบการชนของวัตถุในเกม เช่น ถ้าเก็บไอเทมก็จะได้รับ HP เป็นต้น

Prefab หมายถึง การสร้าง GameObject ตัวต้นแบบ โดยทำการโคลนนิ่งวัตถุที่มีคุณสมบัติต่างๆที่ต้องการ เพื่อนำวัตถุเหล่านั้นมาใช้งานซ้ำโดยที่ไม่ต้องสร้างหรือกำหนดคุณสมบัติหลายครั้ง



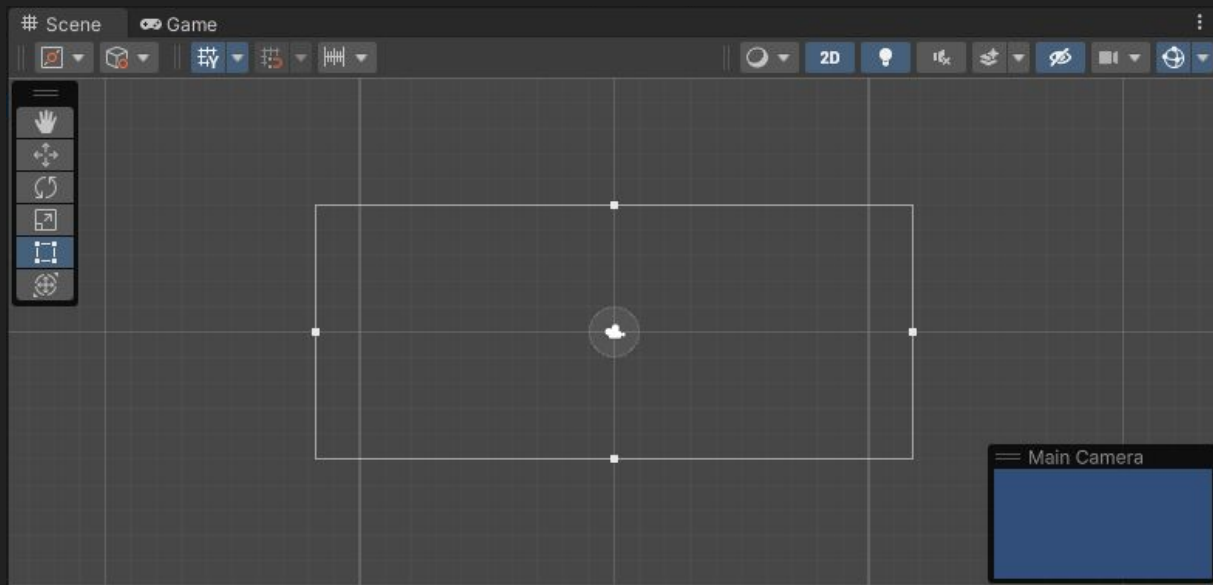
คำศัพท์พื้นฐาน

Tag หมายถึง ป้ายกำกับที่ใช้แบ่งประเภทของวัตถุ เช่น Player , Camera , Enemy , NPC, Ground เป็นต้น

Layer หมายถึง ลำดับชั้นการทำงานของวัตถุ

Scene View เบื้องต้น

Scene View เบื้องต้น



Scene View เบื้องต้น

คำสั่ง	คำอธิบาย
เมาส์ซ้าย	ใช้เลือกวัตถุที่อยู่ใน Scene View
เมาส์ขวา	คลิกข้างไว้เพื่อหมุนดูวัตถุโดยรอบ
เมาส์กลาง (Scroll)	เลื่อนดูมุมมองแบบอิสระ (Hand Tool)
Scroll เมาส์ขึ้น	ซูมเข้า
Scroll เมาส์ลง	ซูมออก
Double Click	เลือกวัตถุแล้วซูมเข้า



Scene View เบื้องต้น

คำสั่ง	คำอธิบาย
Arrow (ลูกศร)	ควบคุมมุมมองแสดงผลตามทิศทางที่ต้องการ
Number 2	ปรับการแสดงผลแบบ 2 มิติ หรือ 3 มิติ
W (Move Tool)	ควบคุมการเคลื่อนที่วัตถุในแกนที่ต้องการ
E (Rotate Tool)	หมุนวัตถุรอบแกนที่ต้องการ
R (Scale)	ย่อ - ขยายวัตถุตามแกนที่ต้องการ



การสร้าง C# Script

รูปแบบสร้าง C# Script

1. Project > Create > C# Script > ตั้งชื่อ .cs
2. คลิกขวา > Create > C# Script > ตั้งชื่อ .cs
3. Assets > Create > C# Script > ตั้งชื่อ.cs

ข้อควรระวังในการทำงานกับไฟล์ Script

1. ควรตั้งชื่อไฟล์ Script ให้เสร็จก่อนเพื่อให้ระบบนำชื่อไปกำหนดเป็นชื่อ Class
2. ชื่อไฟล์ Script กับชื่อ Class ต้องเป็นชื่อเดียวกันเท่านั้น
3. ควรกำหนดอักขรตัวแรกของไฟล์ Script เป็นตัวพิมพ์ใหญ่
4. ถ้าไม่ตั้งชื่อ Script ในตอนแรกที่เราสร้าง แล้วมาแก้ไขชื่อ Class ในภายหลัง อาจจะส่งผลให้ Script นั้นไม่สามารถทำงานได้
5. ควรเลี่ยงการตั้งชื่อไฟล์ Script เป็นภาษาไทย



วิธีการนำไฟล์ Script ไปทำงาน

1. เลือกวัตถุที่ต้องการนำ Script ไปรัน > ลาก Script ไฟล์ (.cs) ไปใส่ในส่วนของ Inspector วัตถุ
2. เลือกวัตถุที่ต้องการนำ Script ไปรัน > ไปที่ Inspector > Add Component > Scripts > ชื่อไฟล์.cs

รู้จักกับฟังก์ชัน Start และ Update



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

ฟังก์ชัน Start และ Update

Start() คือ ฟังก์ชันหรือเมธอดที่จะถูกเรียกใช้งานในตอนเริ่มต้น โดยคำสั่งที่เขียนภายใน Start จะถูกเรียกใช้งานแค่ครั้งเดียวและทำงานเมื่อสั่งให้ Script Enabled เท่านั้น

ฟังก์ชัน Start และ Update

Update() คือ ฟังก์ชันหรือเมธอดที่ใช้รันคำสั่งเป็น Frame ต่อ Frame คือ ทำงานตลอดเวลาโดยอ้างอิงตามความเร็วของเครื่องคอมพิวเตอร์ที่ใช้รันเกม

ฟังก์ชัน Start และ Update

Update() จะทำงานและอ้างอิงตามการคำนวณค่า Frame

Rate (FPS : Frame Per Second)

- ถ้า FPS ต่ำก็จะทำงานช้า
- ถ้า FPS สูงก็จะทำงานเร็ว

เช่น FPS = 60 หมายถึง ทำงาน 60 Frame ต่อวินาที นั่นเอง

รู้จักกับฟังก์ชัน
Start และ Awake

ฟังก์ชัน Start และ Awake

1. **Start()** คือ ฟังก์ชันหรือเมธอดที่จะถูกเรียกใช้งานในตอนเริ่มต้นโดยคำสั่งที่เขียนภายใน Start จะถูกเรียกใช้งานแค่ครั้งเดียวและทำงานเมื่อสั่งให้ Script Enabled เท่านั้น
2. **Awake()** มีการทำงานคล้ายกับ Start คือทำงานในตอนเริ่มต้น แต่ Awake จะทำงานเมื่อโหลด Component หรือ Script เข้ามาทำงาน คือสั่ง Enabled / Disabled ก็ทำงานเหมือนเดิม



ฟังก์ชัน

FixedUpdate & LateUpdate

ฟังก์ชัน Fixed Update

FixedUpdate() ทำงานคล้ายๆกับ Update แต่ว่าจะมีการคำนวณระยะเวลาคงที่ (ค่าตายตัว) และ Realtime ไม่ได้จำแนกตามความเร็วเครื่องใช้ร่วมกับระบบเวลาและระบบฟิสิกส์ (**FrameRate คงที่**)

ยกตัวอย่าง เช่น ถ้าใช้ Update นานๆจะมีการ กระตุกเนื่องจากอ้างอิงตามความเร็วเครื่อง ส่วน FixedUpdate จะคงที่



ฟังก์ชัน LateUpdate

ฟังก์ชัน `LateUpdate()` จะทำงานหลังฟังก์ชัน `Update` อีกที
ใช้ทำงานคำสั่ง เมื่อ Object อัปเดตค่าเรียบร้อยแล้ว

`Time.deltaTime` คำนวณความต่างเวลาระหว่าง Frame ปัจจุบัน
กับ Frame ก่อนหน้า

รับค่าจากแป้นพิมพ์

รับค่าจากแป้นพิมพ์ (Keyboard)

คำสั่งที่ใช้รับค่า `Input.GetAxis ()`

ชื่อ **Axis : Horizontal** เคลื่อนที่ในแนวนอน

ควบคุมการทำงานด้วย

- ปุ่ม Left Arrow , Right Arrow
- ปุ่ม A , D



เคลื่อนที่วัตถุด้วย Rigidbody2D

เคลื่อนที่วัตถุด้วย Rigidbody2D

คำสั่งที่ทำให้วัตถุเคลื่อนที่ได้

- **Rigidbody2D.velocity** ทำให้วัตถุเคลื่อนที่ในแกนที่ต้องการด้วยความเร็วคงที่

AddForce

AddForce

- **RigidBody2D.AddForce** คือ การทำให้วัตถุเคลื่อนที่โดยอาศัยการเพิ่มแรงเข้าไปโดยวัตถุจะเคลื่อนที่ด้วยความเร็วสูงสุดในตอนเริ่มต้นแล้วลดความเร็วลงเมื่อเวลาผ่านไป (คิดแรงเสียดทาน)

AddForce

AddForce จะมีความเร็วสัมพันธ์กับขนาดมวลของวัตถุโดยอ้างอิงจาก
สมการแรงและการเคลื่อนที่ของนิวตัน

$$F=ma$$

มวลน้อย = ใช้แรงน้อย | มวลมาก = ใช้แรงมาก



รับค่าจากแป้นพิมพ์ (Keyboard)

คำสั่งที่ใช้รับค่า

- ชื่อ **Axis : Jump** เคลื่อนที่ในแนวตั้ง
- ควบคุมการทำงานด้วยปุ่ม SpaceBar



การสร้าง Prefab

โคลนนิ่งวัตถุด้วย Prefab

Prefab หมายถึง การสร้าง GameObject ตัวต้นแบบ
โดยทำการโคลนนิ่งวัตถุที่มีคุณสมบัติต่างๆที่ต้องการเพื่อจะ
นำวัตถุเหล่านั้นมาใช้งานซ้ำโดยที่ไม่ต้องสร้างหรือกำหนด
คุณสมบัติหลายครั้ง

การชนของวัตถุ (Collision)

ขอบเขตการชน (Collider2D)

การชนของ GameObject จะอาศัยส่วนที่เรียกว่า
“ **Collider2D** ” โดยขอบเขตการชนจะขึ้นอยู่กับรูปแบบ
ของ Collider2D ที่ใช้ เช่น

- สี่เหลี่ยม (BoxCollider2D)
- วงกลม (CircleCollider2D)



ประเภทของการชน

- การชนแบบไม่ทะลุ
- การชนแบบทะลุ (Trigger)

ป้ายกำกับ (Tag)

Tag คืออะไร

- Tag คือ ป้ายกำกับที่ใช้แบ่งประเภทของวัตถุ เช่น Player Camera , Enemy , NPC, Ground เป็นต้น
- สามารถนำ Tag มาใช้ตรวจสอบการทำงานกับวัตถุที่สนใจได้ เช่น ตรวจสอบการชนของวัตถุ



ใช้การชนด้วย Collision2D

ใช้การชนด้วย Collision2D

สำหรับตรวจสอบการชนของวัตถุที่ไม่สามารถทะลุได้
โดยใช้คำสั่ง 2 ดังนี้

- OnCollisionEnter2D สำหรับตรวจสอบว่าวัตถุ
อยู่ในขอบเขตการชนหรือไม่
- OnCollisionExit2D สำหรับตรวจสอบว่าวัตถุ
อยู่นอกขอบเขตการชนหรือไม่

ใช้คอนโทรลด้วย
OnTrigger2D

ใช้การชนด้วย OnTriggerEnter2D

สำหรับตรวจสอบการชนของวัตถุที่สามารถทะลุได้
โดยใช้คำสั่ง 2 ดังนี้

- OnTriggerEnter2D สำหรับตรวจสอบว่าวัตถุ
อยู่ในขอบเขตการชนหรือไม่
- OnTriggerExit2D สำหรับตรวจสอบว่าวัตถุ
อยู่นอกขอบเขตการชนหรือไม่

คอร์สสร้างเกม 2D ด้วยโปรแกรม Unity

วิทยากร

Kong Ruksiam

Programmer , Developer

ผู้เรียนทั้งหมด รีวิว


1,912 **450**

เกี่ยวกับฉัน

โปรแกรมเมอร์และนักพัฒนาเกม รวมถึงสอนเกี่ยวกับการเขียนโปรแกรมในช่องยูทูป KongRuksiam , KongRuksiam Official และเป็นเจ้าของแฟนเพจ KongRuksiam มีผู้ติดตามมากกว่า 50,000 คน



 Facebook

 Youtube

<https://www.udemy.com/course/unity-2d-tutorial/>