3D Game Programming
2D primitive

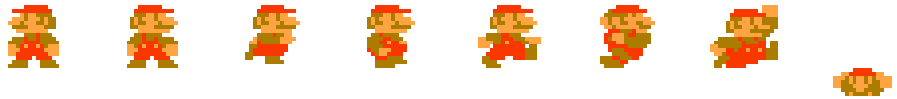
Ming-Te Chi
Department of Computer Science,
National Chengchi University

Outline

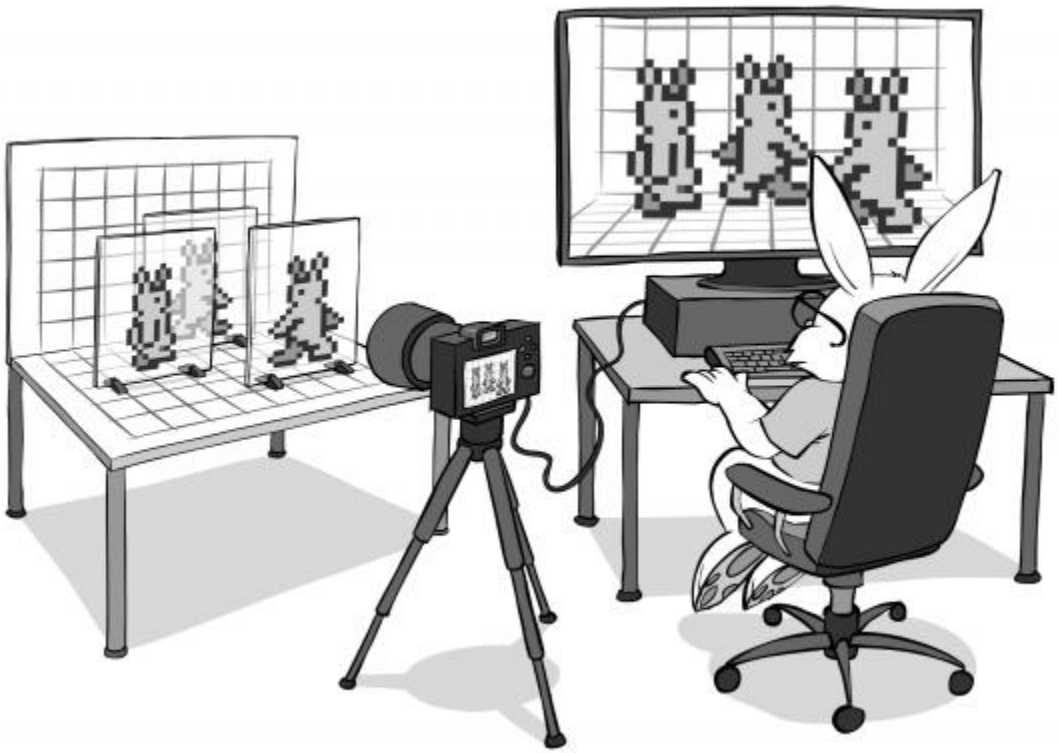
- Imaging and Raster Primitives
  - Deformation and skeletal Animation
  - Alpha and Blending
  - Intersection
- Object-Oriented Programming
- Game LOOP Concept
- Physics 2D
  - Rigid body
  - Collider and trigger

IMAGING AND RASTER PRIMITIVES

Sprite

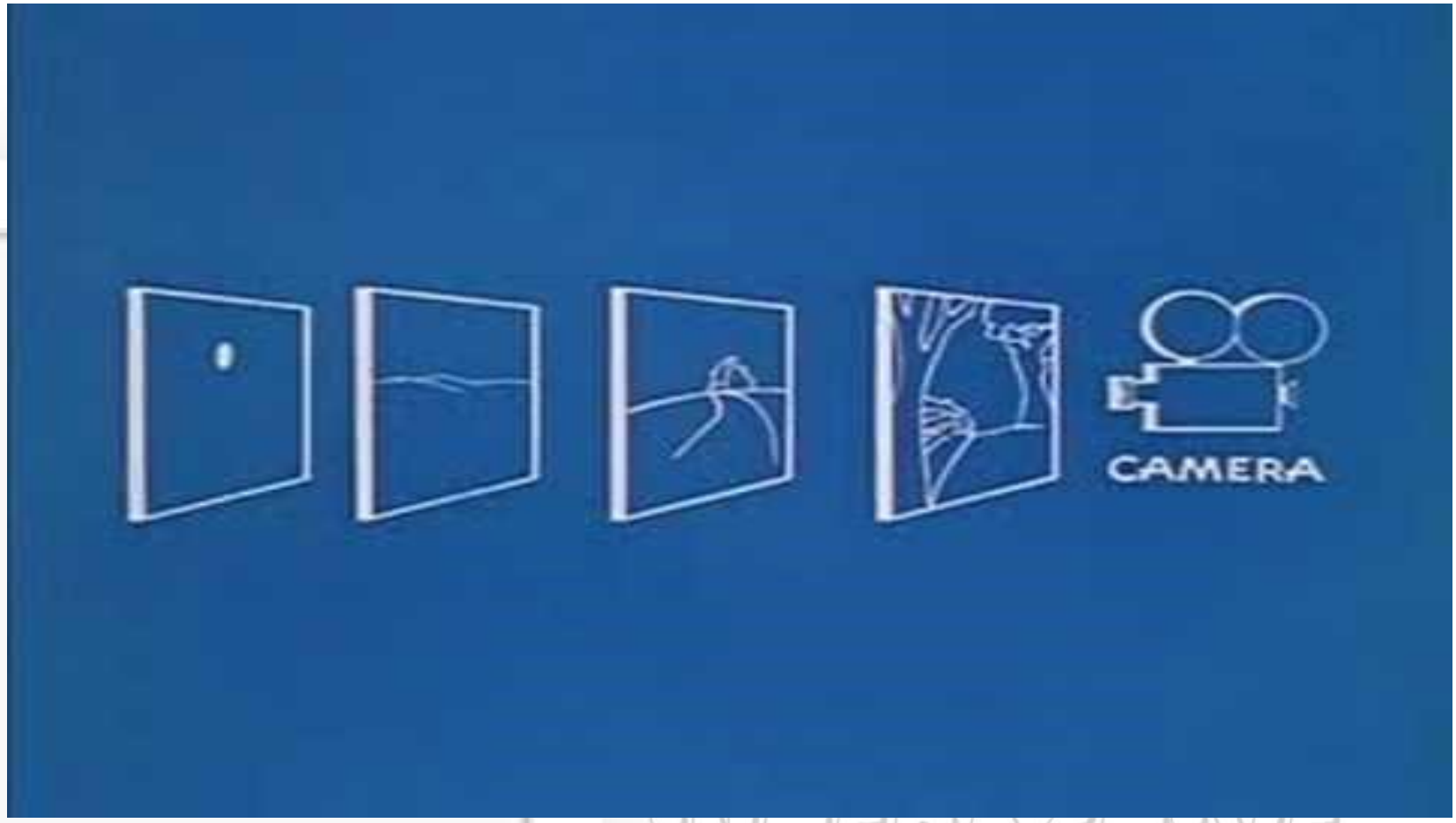


Super Mario Bros. Nintendo



http://ithare.com/category/distributed-systems/

Walt Disney's MultiPlane Camera (Filmed: Feb. 13, 1957)



Rayforce 1994

Electromagnetic spectrum

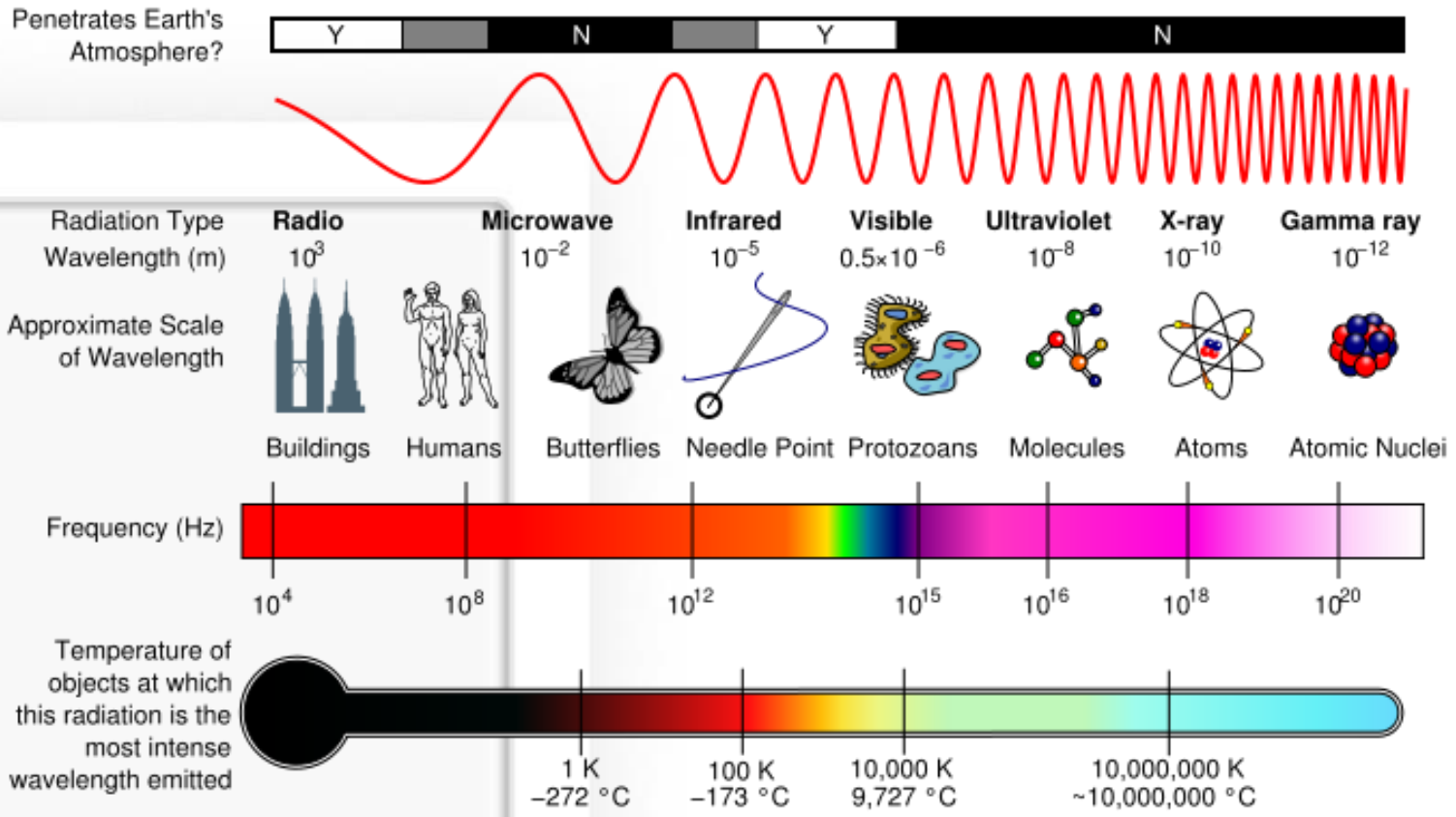
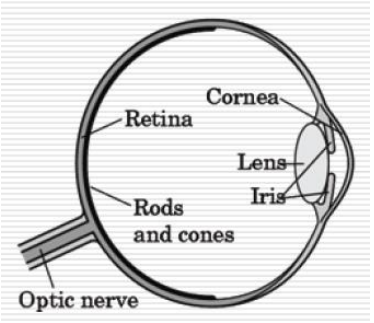
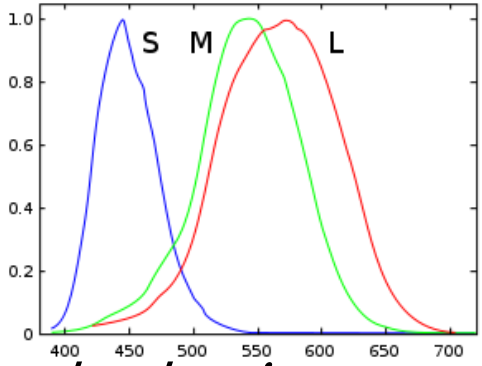


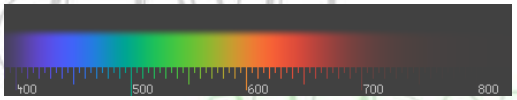
Image from wiki

Three-Color Theory

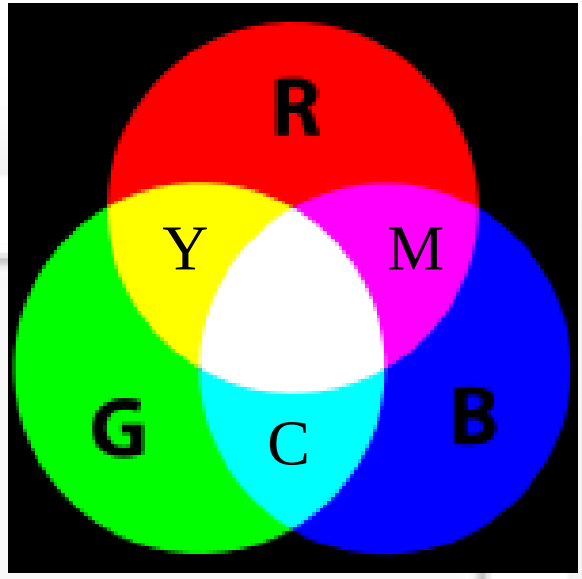
- Human visual system has two types of sensors
  - Rods:
    - monochromatic, night vision
  - Cones:
    - Color sensitive
    - Three types of cone
    - Only three values
    - (the *tristimulusvalues) are sent to the brain*





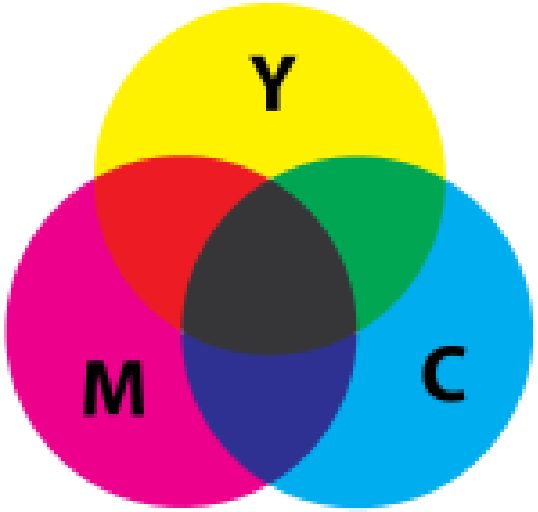


Additive / Subtractive color



Additive Color

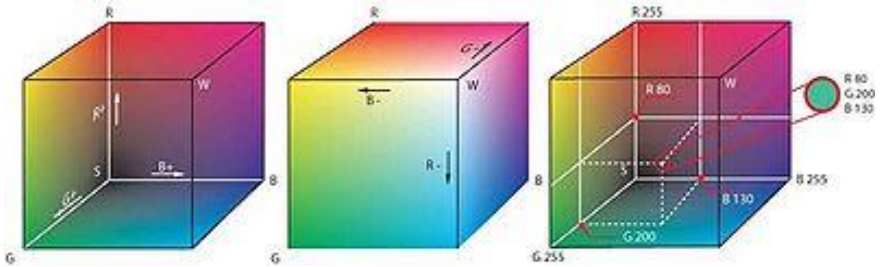
LCD, projector

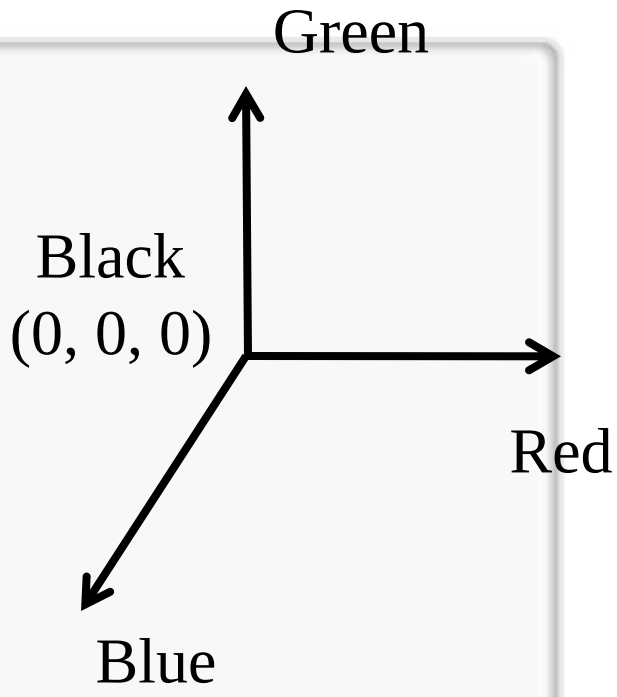


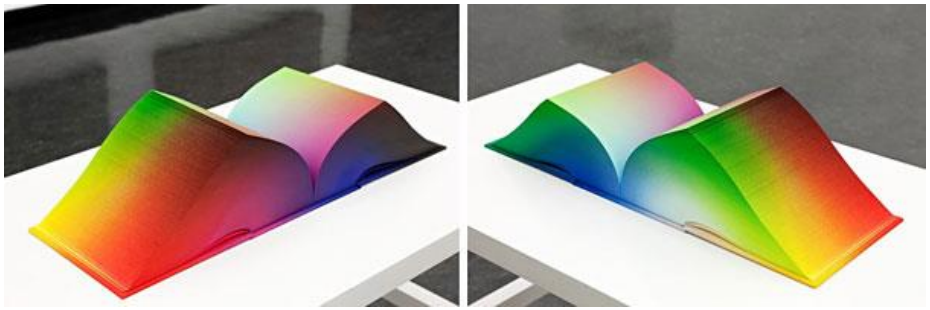
Subtractive Color

Printer

RGB color space

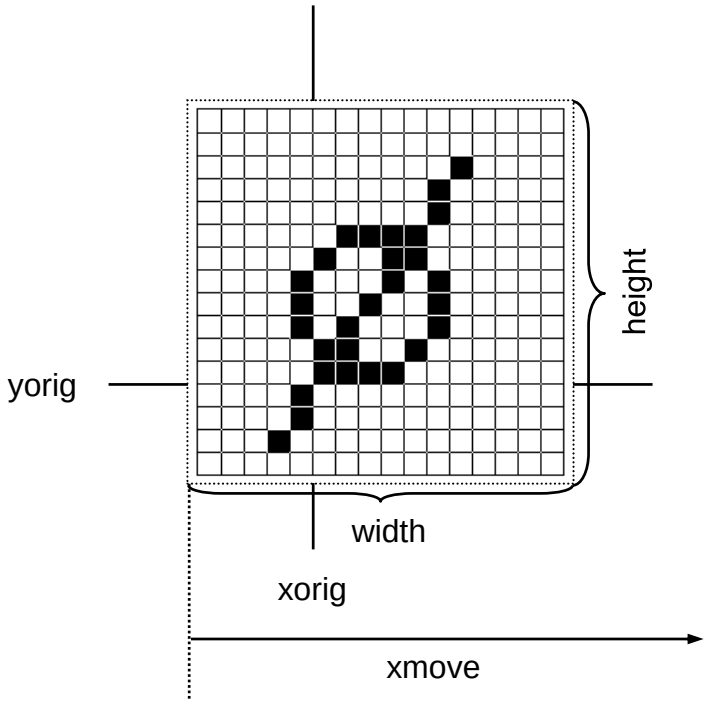






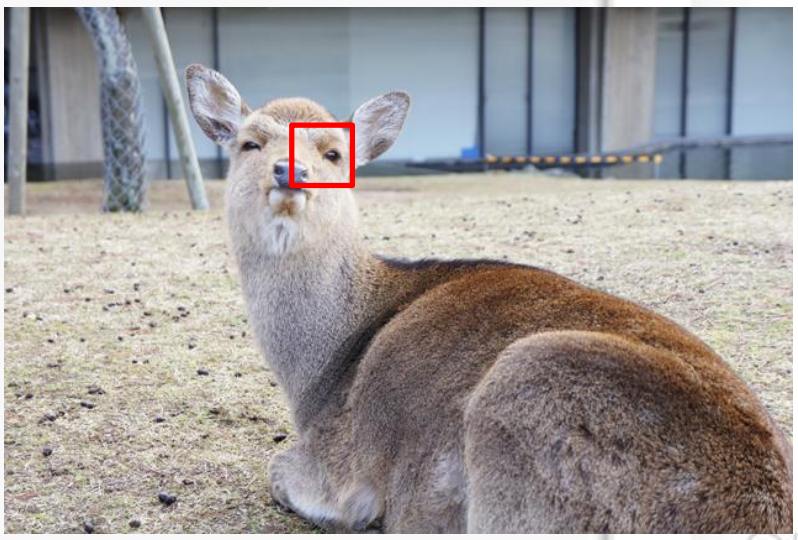
a 8 x 8 x 8-inch cube book
Tauba Auerbach

Bitmap



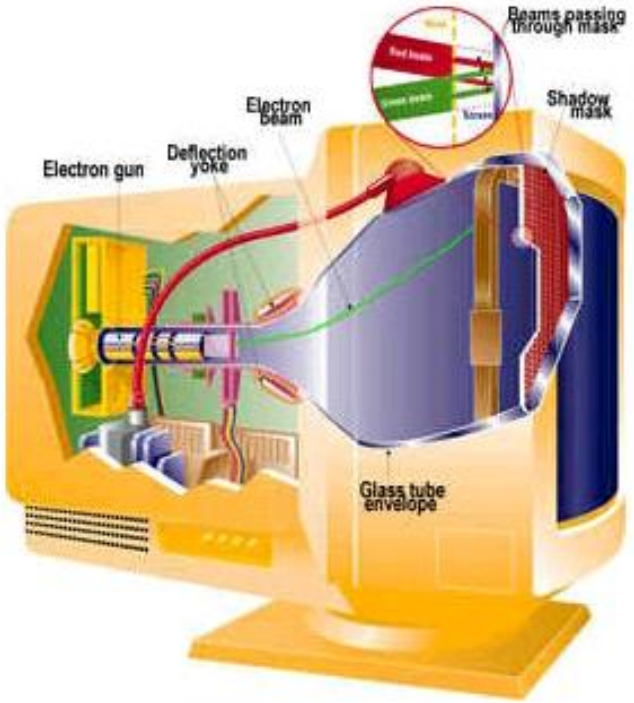
Raster Graphics

- Image produced as an array (the *raster*) of picture elements (*pixels*) in the *frame buffer*

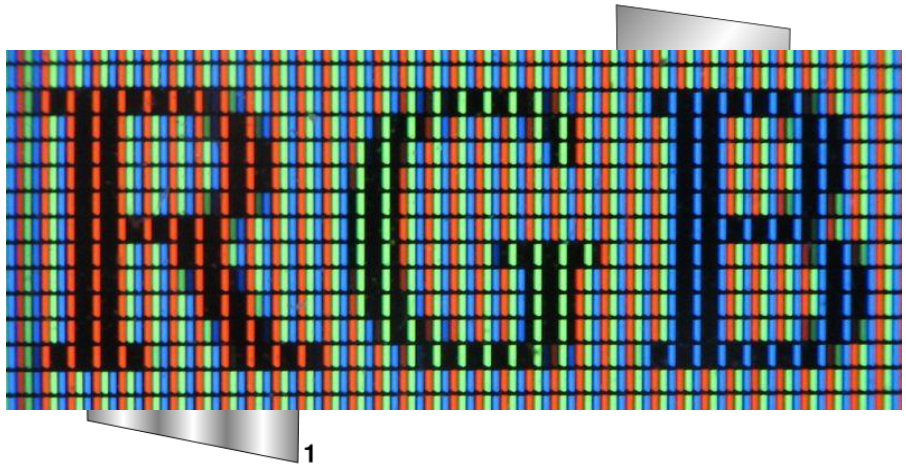


(179, 161, 153)

Display Technologies



CRT



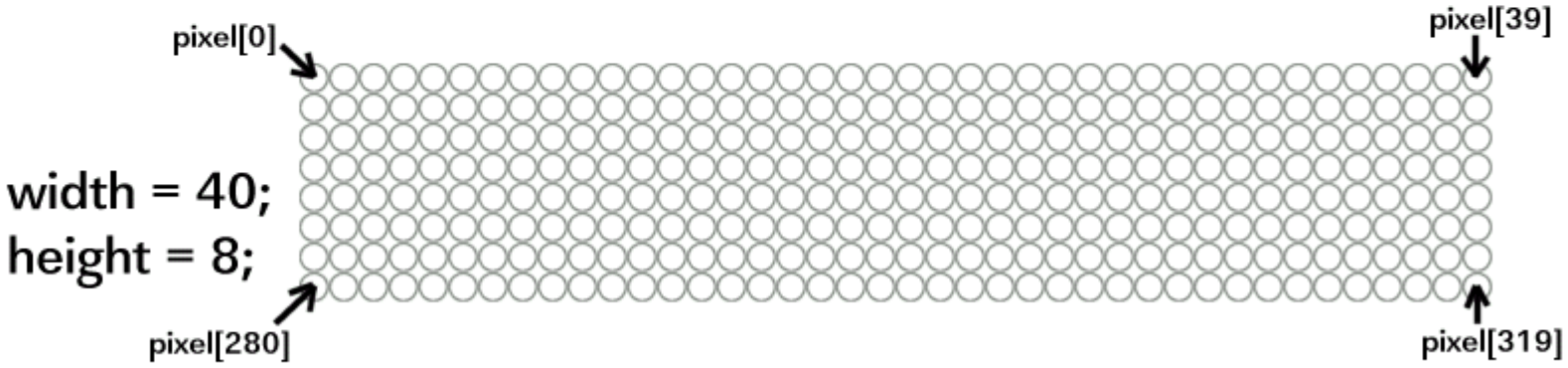
LCD

Lets Talk About Pixels

- Pixels are stored as a 1-dimensional array of *int*s
- Each *int* is formatted according to Java's standard pixel model

The 4 bytes of a 32-bit *Pixel* int.
if Alpha is 0 the pixel is transparent.
if Alpha is 255 the pixel is opaque.

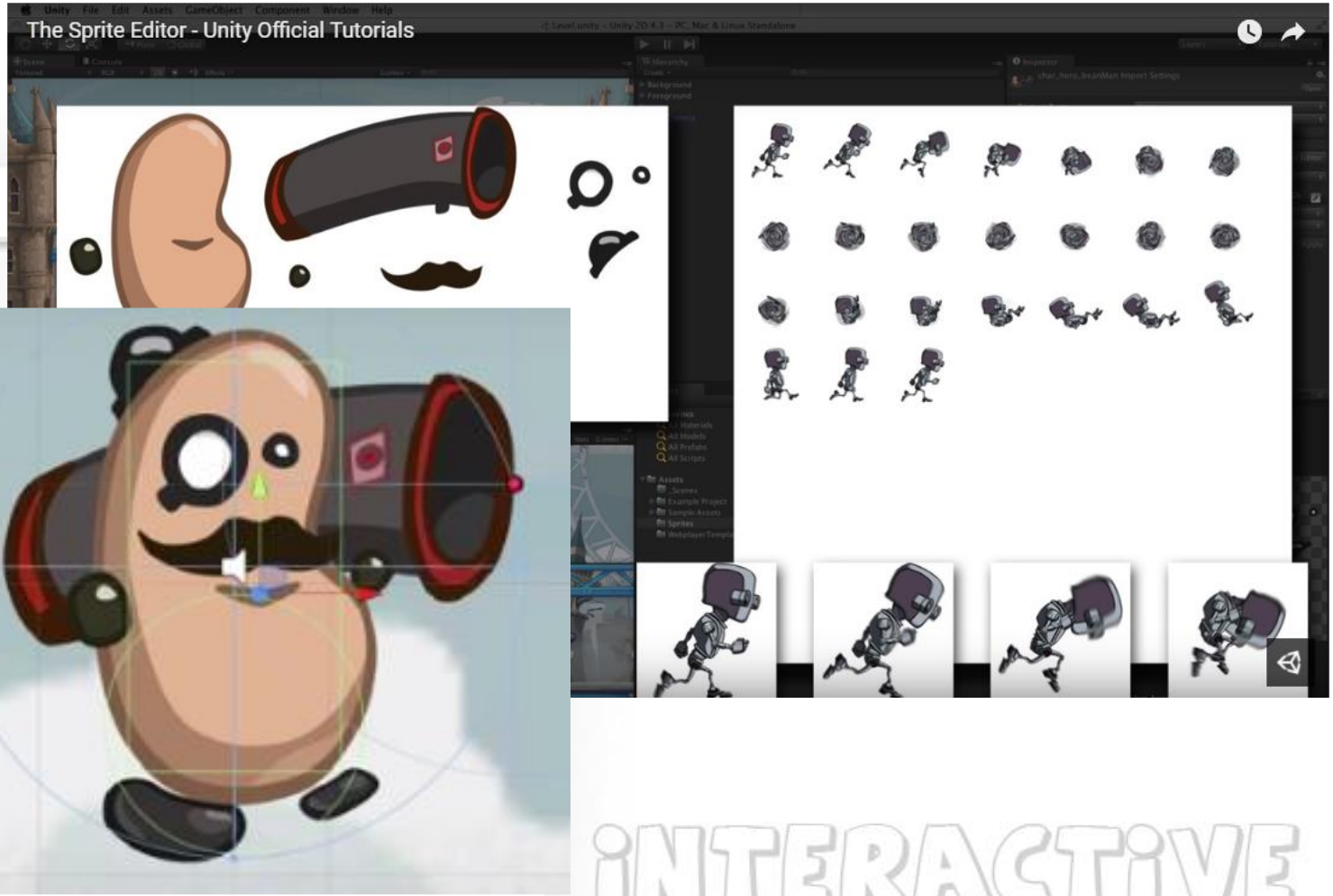
- Layout of the pixel array on the display:



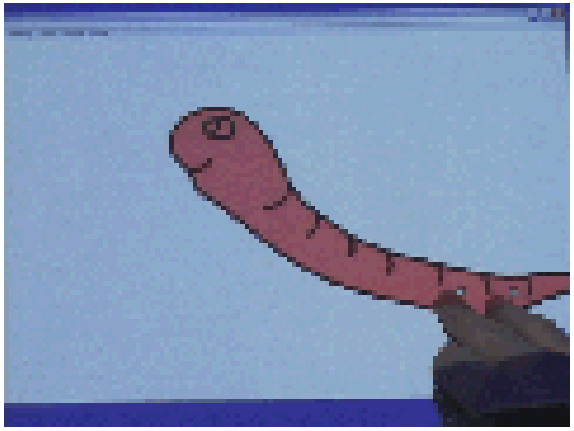
- This is the image format used internally by Java

ANIMATION

Component, Sprite

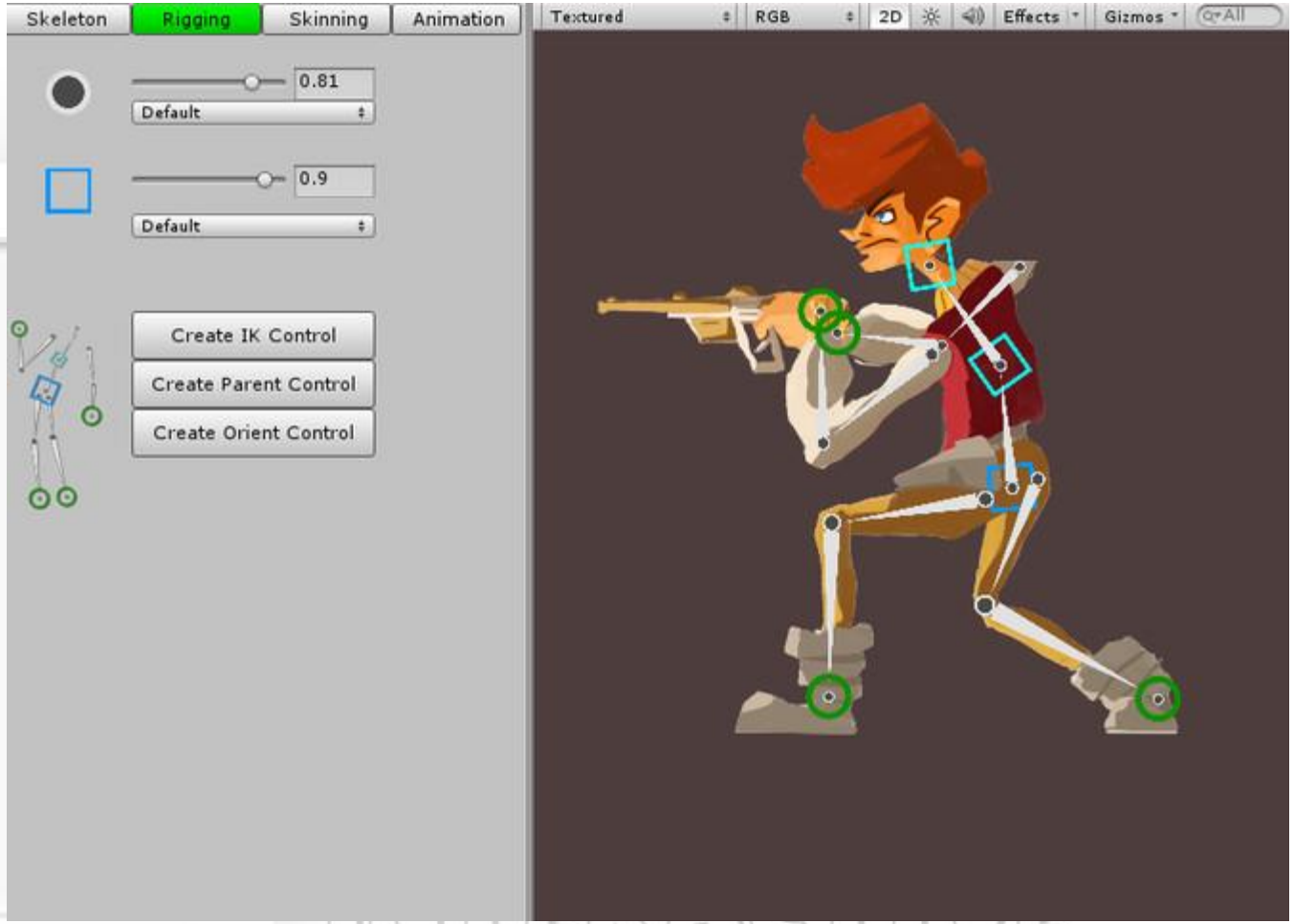


Deformation



As-Rigid-As-Possible Shape Manipulation

Skeletal animation

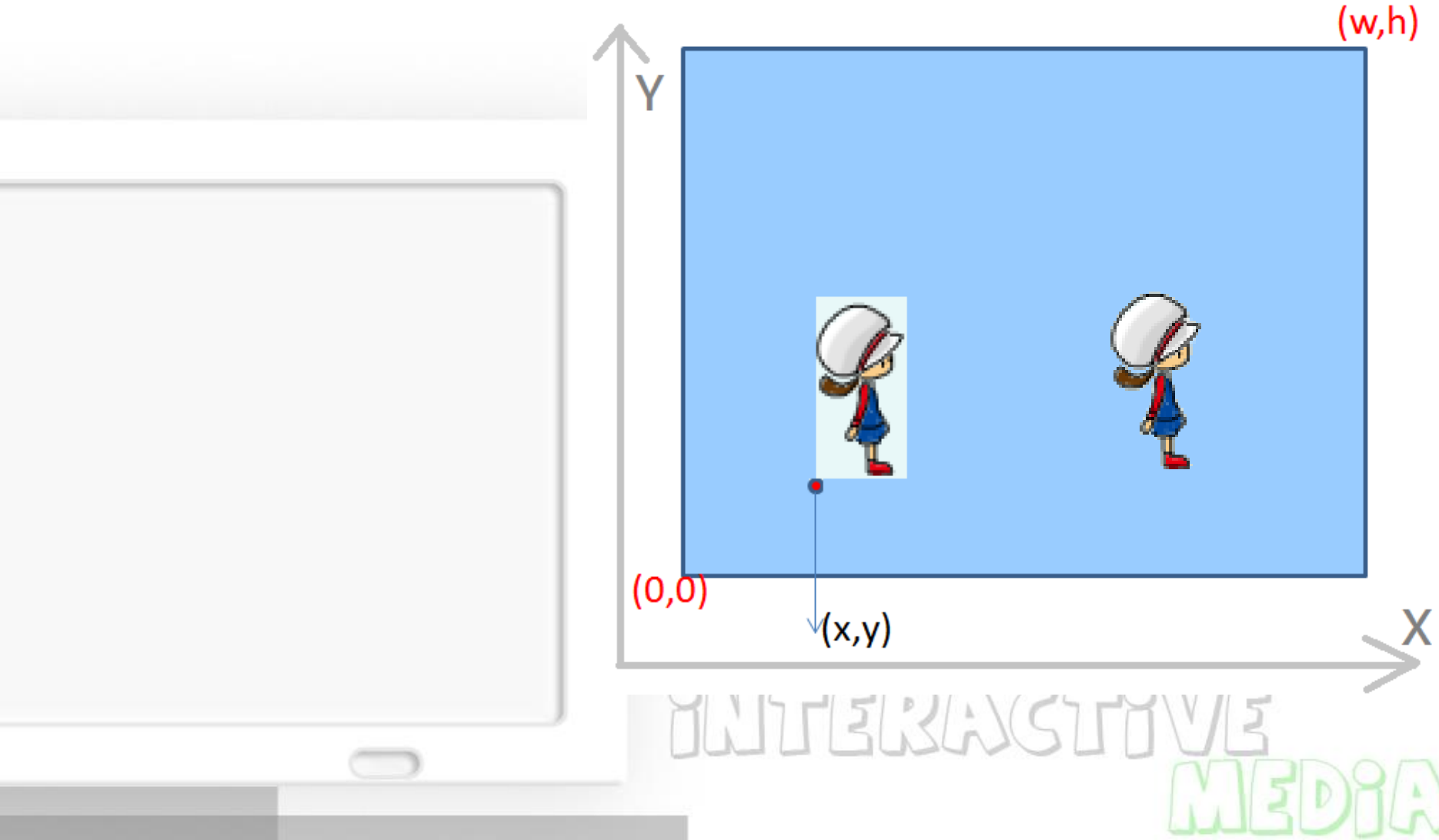


Puppet2D

Live2D

ALPHA AND BLENDING

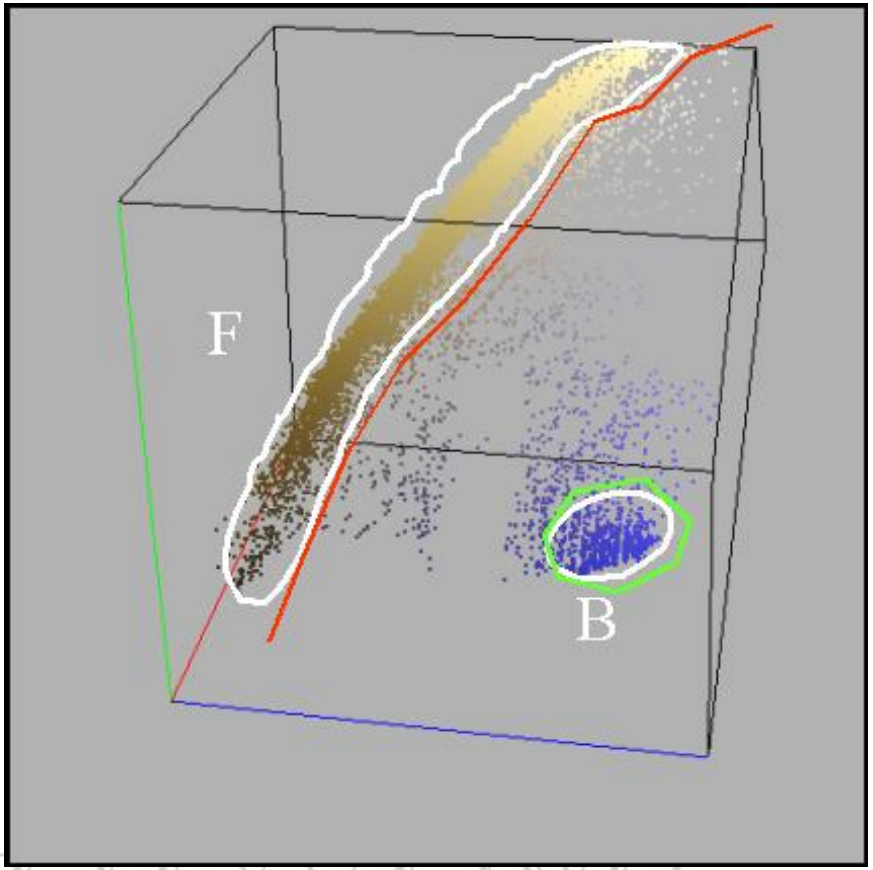
Draw image



Alpha

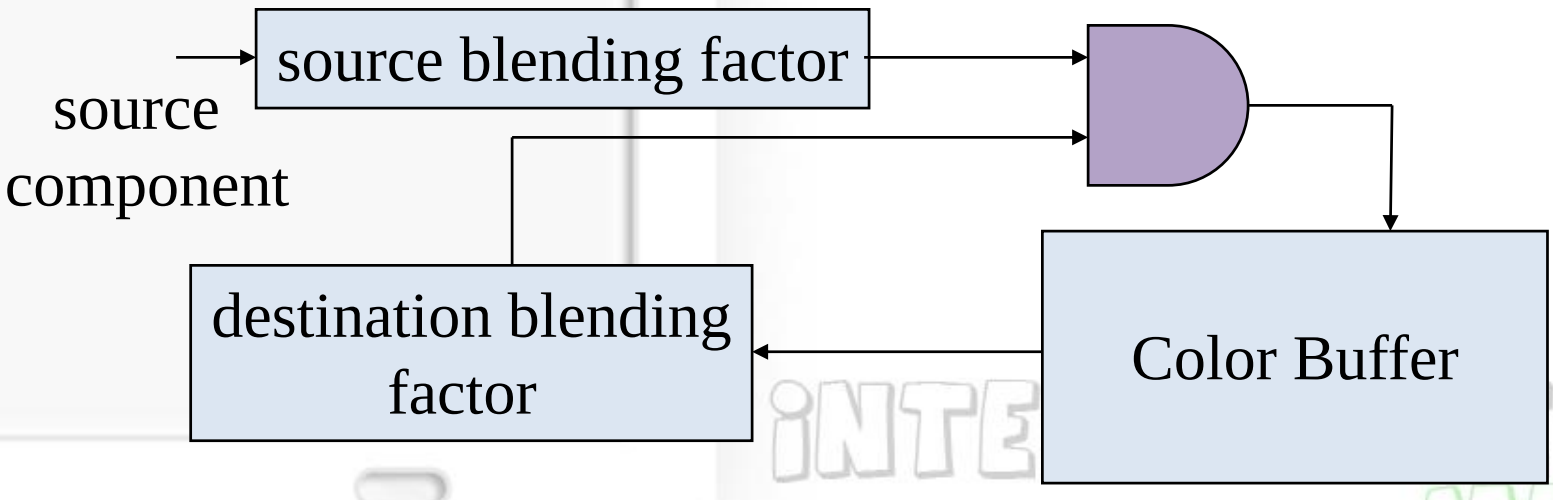
- An alpha channel, representing transparency information on a per-pixel basis.
- Alpha = 0.0f: fully transparent
- Alpha = 1.0f: fully opaque

Chroma-keying (Primatte)



Writing Model

- Use A component of RGBA (or RGBα) color to store opacity
- During rendering we can expand our writing model to use RGBA values



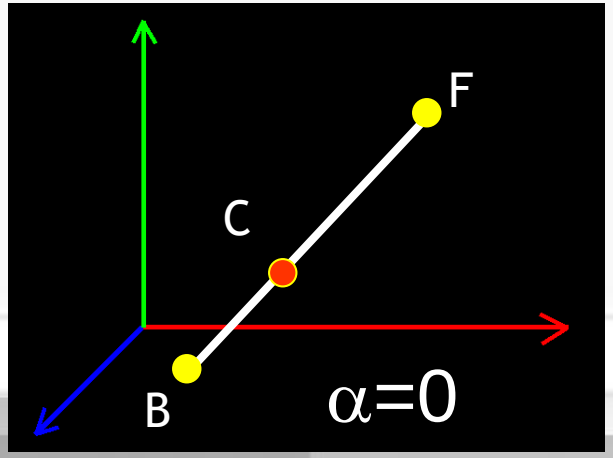
Blending

- glBlendFunc(Glenum S, Glenum D);
  - Cf = (Cs*S) + (Cd*D)
- glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

Ex: Cs={Rs, Gs, Bs, As}, Cd ={Rd, Gd, Bd, Ad},
Cf = (Cs*As) + (Cd*(1-As))

Compositing

foreground color alpha matte background



compositing equation

Order Dependency

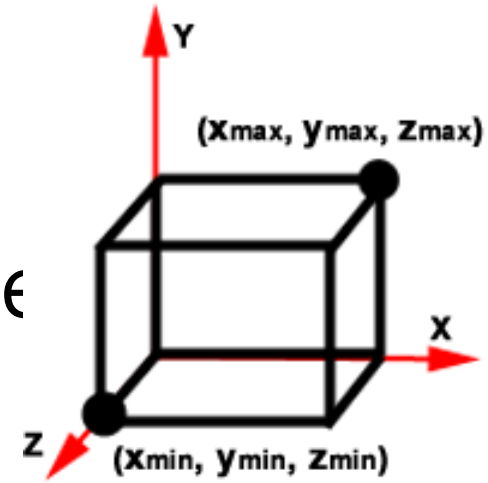
- Is this image correct?
  - Probably not
  - Polygons are rendered in the order they pass down the pipeline
- Blending functions are order dependent

INTERSECTION

Axis-Aligned Bounding Boxes

- Specified as two points:

$$(x_{\min}, y_{\min}, z_{\min}), (x_{\max}, y_{\max}, z_{\max})$$



- Normals are easy to calculate
- Simple point-inside test:

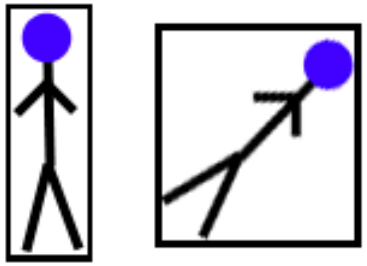
$$x_{\min} \le x \le x_{\max}$$

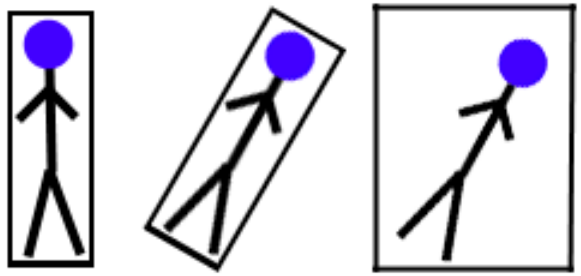
$$y_{\min} \le y \le y_{\max}$$

$$z_{\min} \le z \le z_{\max}$$

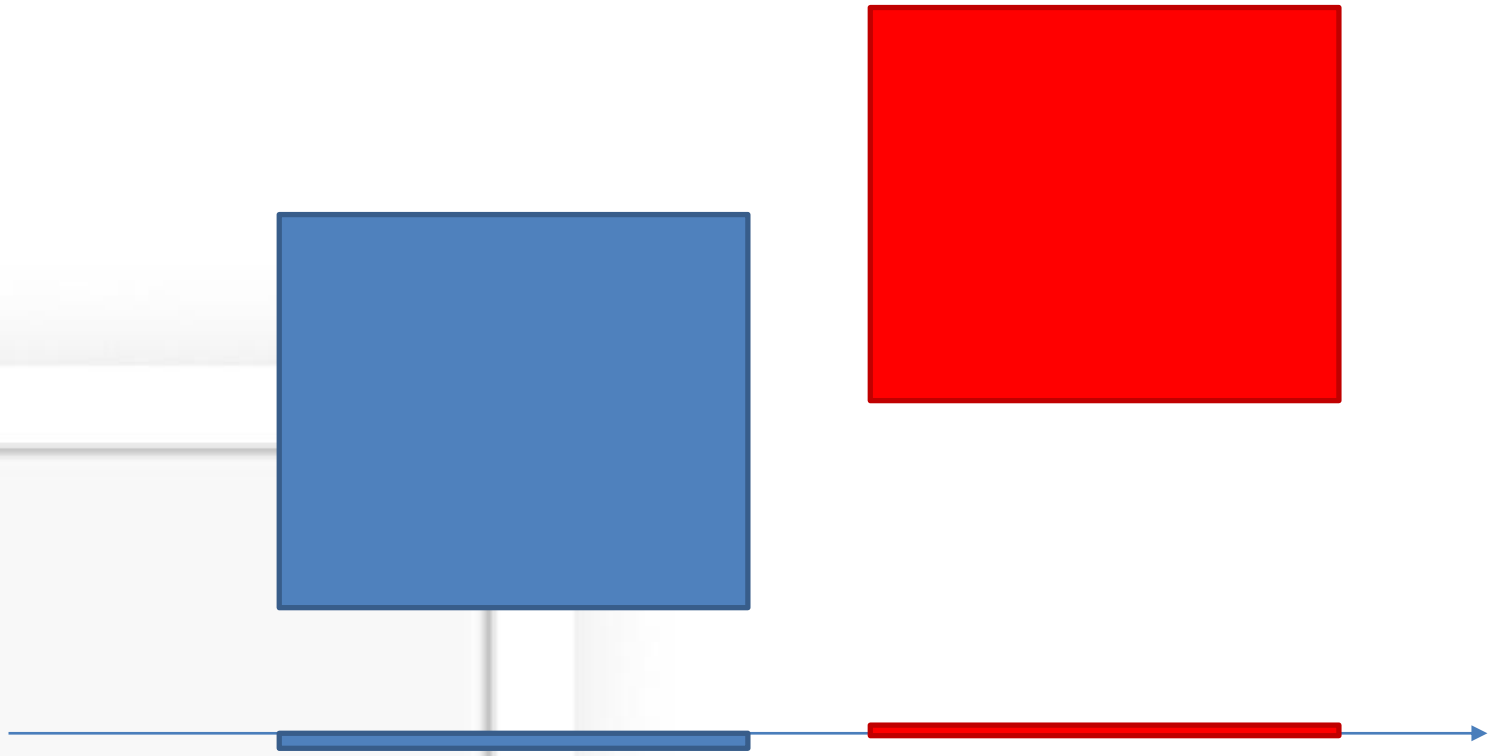
Problems With AABB's

- Not very efficient
- Rotation can be complicated
  - Must rotate all 8 points of box
  - Other option is to rotate model and rebuild AABB, but this is not efficient

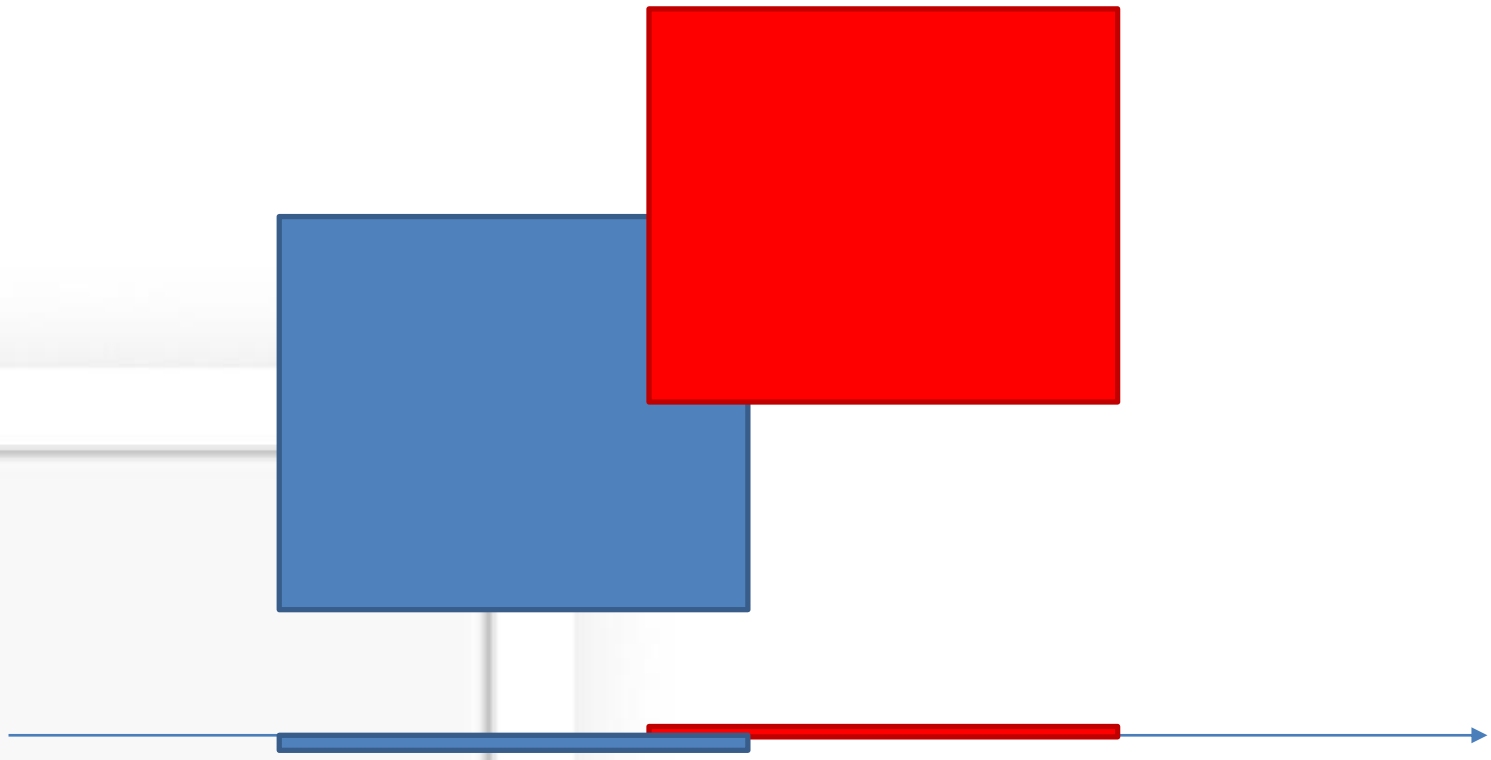





```
function isPointInsideAABB(point, box) {
  return (point.x >= box.minX && point.x <= box.maxX) &&
         (point.y >= box.minY && point.y <= box.maxY);
}
```



```
function intersect(a, b) {
        return (a.minX <= b.maxX && a.maxX >= b.minX) &&
               (a.minY <= b.maxY && a.maxY >= b.minY);
}
```



```
function intersect(a, b) {
        return (a.minX <= b.maxX && a.maxX >= b.minX) &&
               (a.minY <= b.maxY && a.maxY >= b.minY);
}
```

OBJECT-ORIENTED PROGRAMMING

is-a, has-a

- Project
  - Scene1
    - Actor
      - Transform
      - Sprite Renderer
      - Script
      - Box Collider
      - Rigidbody 2D
    - Camera
      - Transform
      - Camera

Actor is a GameObject.

Actor has a Script.

GameObject

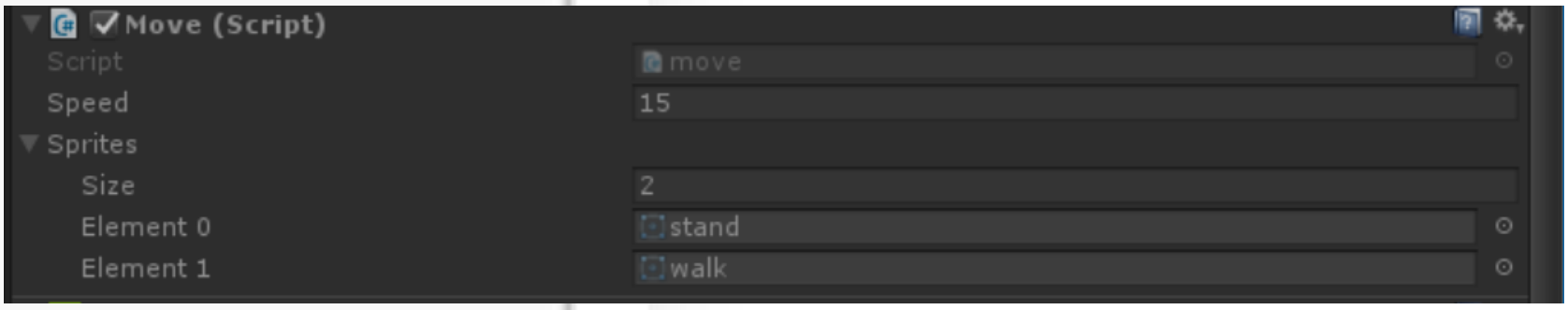
Variables

activeInHierarchy	Is the GameObject active in the scene?
activeSelf	The local active state of this GameObject. (Read Only)
isStatic	Editor only API that specifies if a game object is static.
layer	The layer the game object is in. A layer is in the range [0...31].
scene	Scene that the GameObject is part of.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.

GetComponent<gameManager>()

public

```
public float speed = 20;
public Sprite[] sprites;
int sprites_index = 0;
```



宣告

- public, static
  - int a; //a只會在該腳本被使用、修改
  - public int a; // a可以在editor的介面中直接使用與修改
  - public type function(type argv…) { } //可在其他類別取用
  - public static a; //假設宣告a的腳本為abc.cs，a可以在其他腳本中寫 abc.a 來使用與修改
- Random ← 寫遊戲肯定會用到
  - float a = Random.value; // between 0 ~ 1
  - float a = Random.Range(1.0f , 3.0f);
- Float型態的變數後面的 f 是必須的
  - float a = 2.5; //error
  - float a = 2.5f; //OK

Sprite switching

```
 if (Input.GetKey(KeyCode.D))
 {
this.transform.position += new Vector3(speed *
Time.deltaTime, 0, 0);

this.GetComponent<SpriteRenderer>().sprite =
sprites[(++sprites_index)%2];

}
```

addScore()

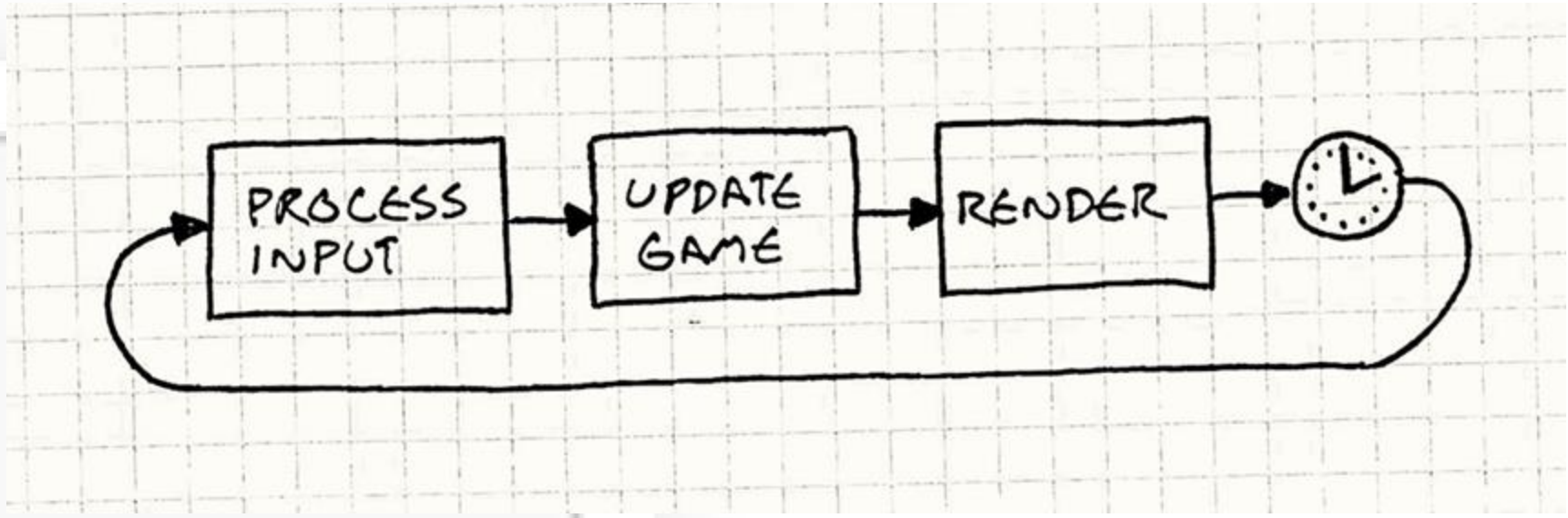
```
public class UFOController: MonoBehaviour {
    …
    void OnTriggerEnter2D(Collider2D other){

GameObject.Find( “GameManager” ).GetComponent<g
ameManager>().addScore(100);

    …
    }
}
```

GAME LOOP

Game loop



http://gameprogrammingpatterns.com/game-loop.html

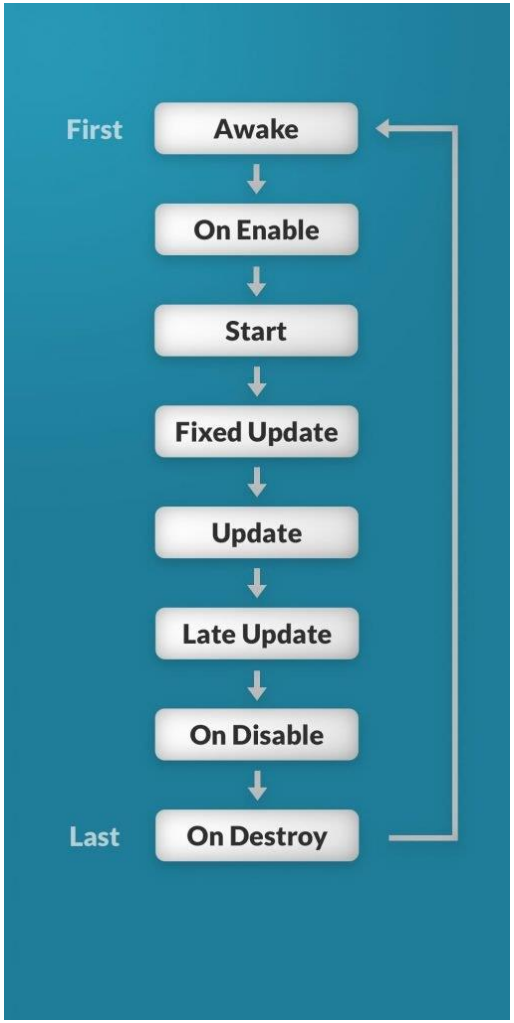
Script-基本函式

函數名	被呼叫時機
Awake()	在脚本(Script)檔被創建並載入場景的時候呼叫。
Start()	在創建後的下一個Update()被呼叫前執行。
Update()	每個禎(frame)更新時呼叫。
FixedUpdate()	在固定的時間呼叫。
OnGUI()	用於處理GUI事件，每個禎(Frame)更新時執行。
OnCollisionEnter()	碰撞體(Collider)/剛體(Rigidbody)和其他的碰撞體(Collider)/剛體(Rigidbody)接觸的一開始執行。
OnCollisionExit()	當碰撞體(Collider)/剛體(Rigidbody)和其他的碰撞體(Collider)/剛體(Rigidbody)終止接觸的瞬間執行。
OnCollisionStay()	當碰撞體(Collider)/剛體(Rigidbody)和其他的碰撞體(Collider)/剛體(Rigidbody)終止接觸的瞬間執行。
OnTriggerEnter()	當本身碰撞體進入觸發器(Trigger)時執行。
OnTriggerExit()	當本身碰撞體離開觸發器(Trigger)的瞬間執行。
OnTriggerStay()	本身碰撞體與觸發器(Trigger)持續接觸的情況下，每個禎(Frame)執行。

Awake VS Start

- Awake：遊戲開始後(play鍵以後)，每一個物件在其active的那一刻就會執行這個函式
(有些物件可能一開始不是active的，在其轉變成active狀態的時候會立即執行awake)

- Start：物件在其active「後」，遊戲進行的下一偵(frame)前會呼叫此函式



https://gamedevbeginner.com/start-vs-awake-in-unity/

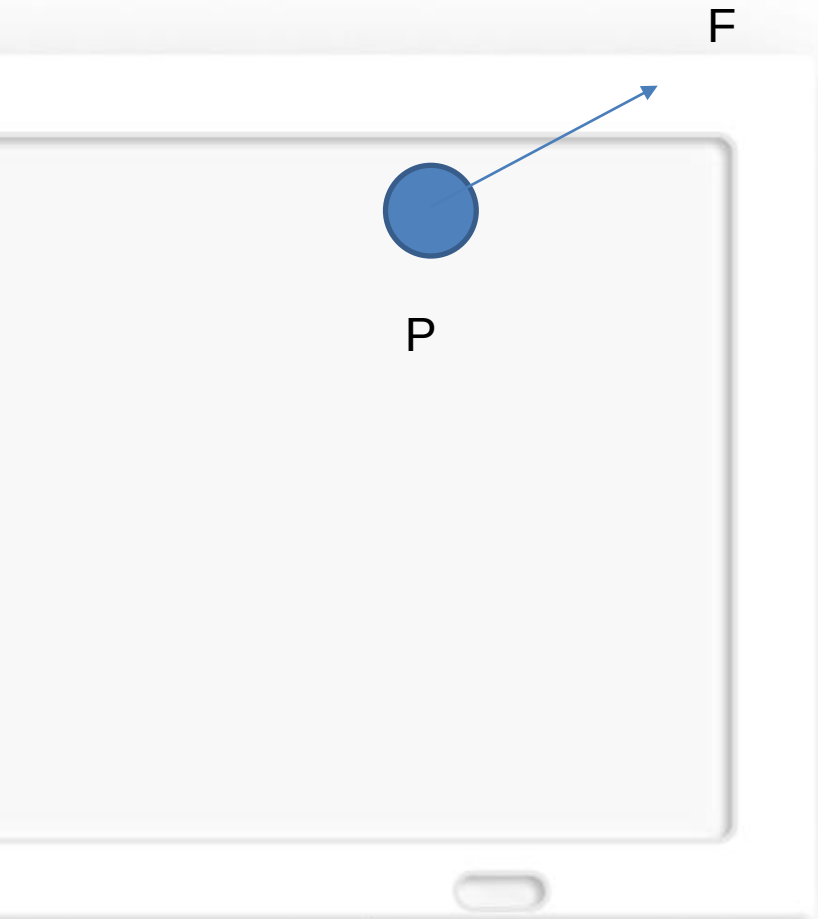
Update() VS FixedUpdate()

- Update：在遊戲進行的每一禎會執行一次(執行的次數受到電腦實際的速度影響)

- FixedUpdate：固定的時間會執行一次，時間可以在Edit→Project settings→Time→Fixed Timestep做更動
  - Fixed Timestep 預設 0.02

PHYSICS 2D

Rigidbody

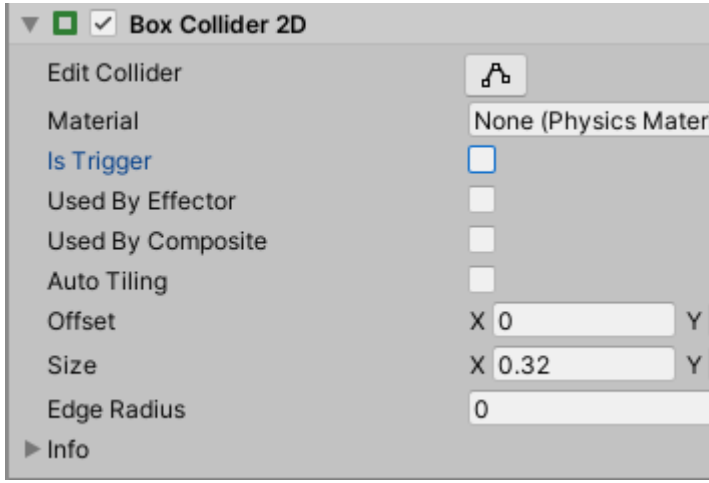


F=ma a=F/m

V= at

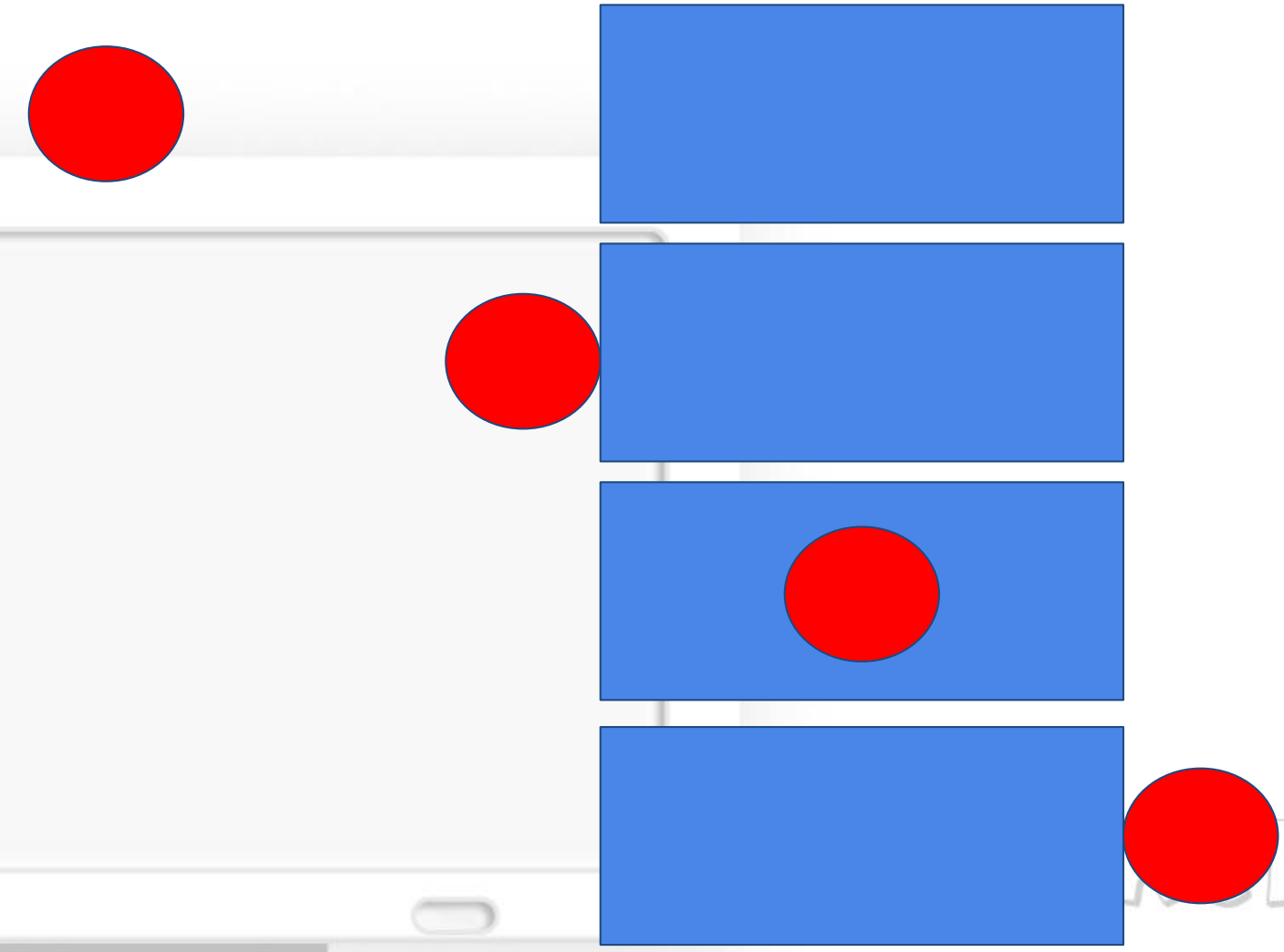
$$Distance = Vt = at^2$$

Collision(碰撞)與Trigger(觸發)



- 碰撞：兩個物件產生物理碰撞，會觸發OnCollisionEnter/Stay/Exit函數
- 觸發：取消所有物理碰撞(勾選is Trigger)，兩物件接觸時會觸發OnTriggerEnter/Stay/Exit
- 換言之兩物件接觸時一定不可能同時觸發Collision和Trigger signal

OnCollisionEnter/Stay/Exit



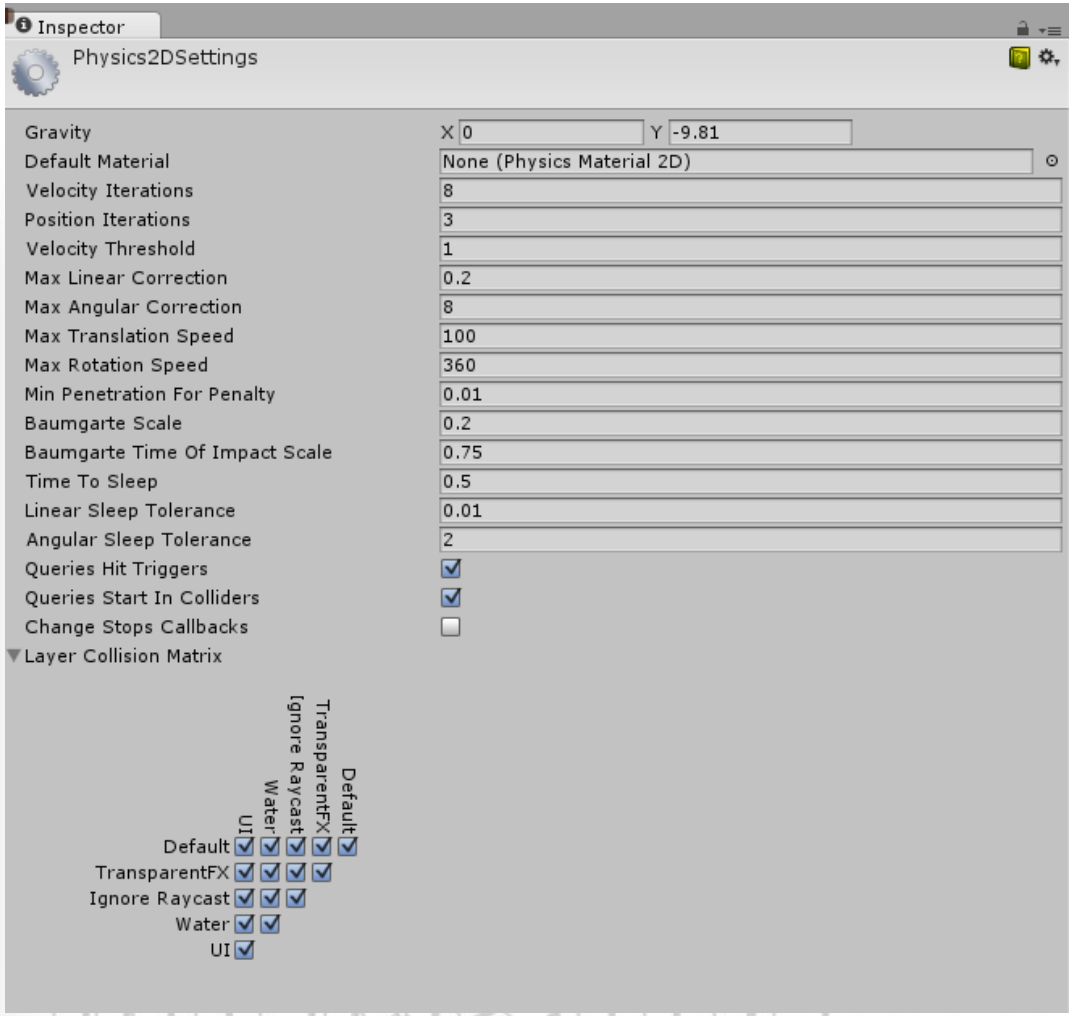
Enter

Stay

Exit

Physics 2D Settings

Edit > Project Settings > Physics 2D



physics2D

static: 未加入rigidbody

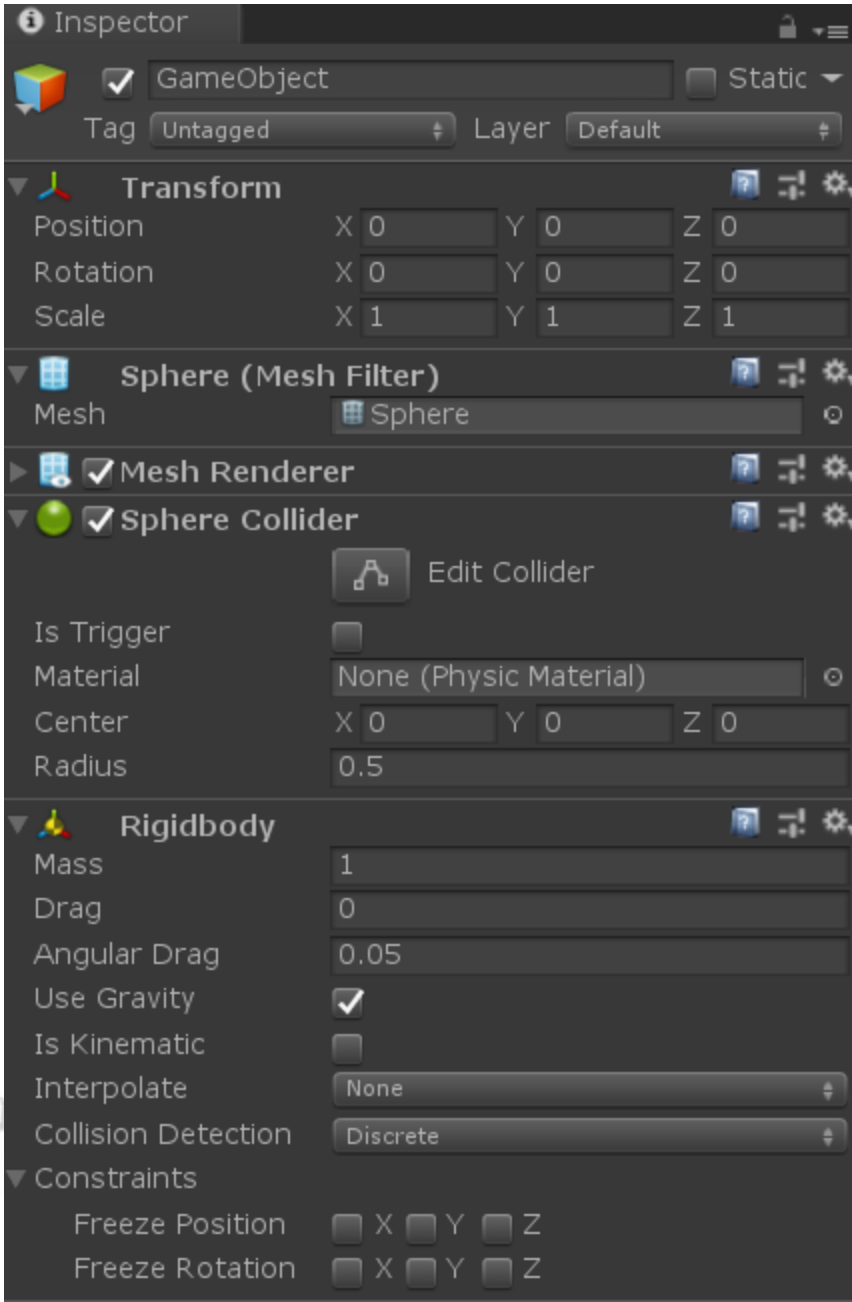
Rigidbody：允許物理運算。Ex: 受力(重力、阻力、慣性)和速度。(add component->Rigidbody)

Kinematic: 不使用物理運算，但可對其他剛體進行物理運算。

(Rigidbody->click 'Is Kinematic')

備忘有reference連結

有興趣自己看>_O



Collision(碰撞)與Trigger(觸發)

- 六種碰撞器的分別

Static Collider Rigidbody Collider Kinematic Rigidbody Collider	Static Trigger Collider Rigidbody Trigger Collider Kinematic Rigidbody Trigger Collider
● 左半邊代表collider在接觸時會產生碰撞訊號	● 右半邊Trigger代表在collider的屬性中勾選了is trigger，使得該碰撞器不會發生碰撞，而會被穿過，被其他物件接觸時產生觸發訊號

Collision action matrix

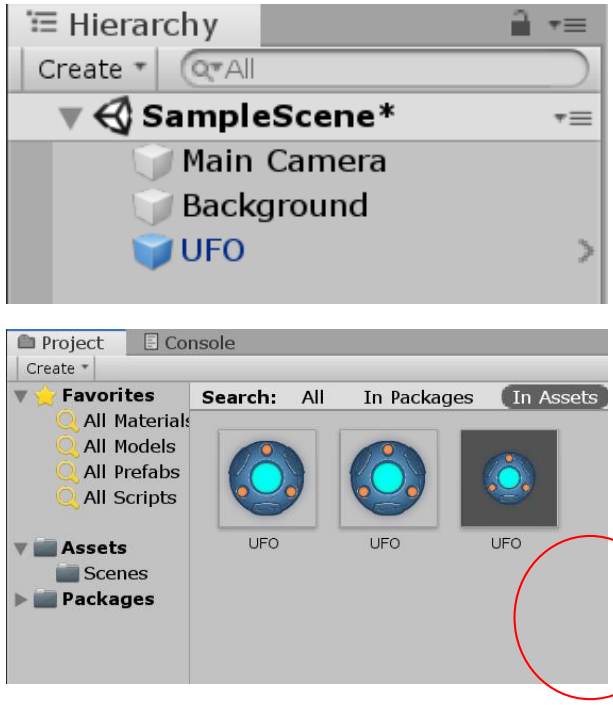
	Static Collider	Rigidbody Collider	Kinematic Rigidbody Collider	Static Trigger Collider	Rigidbody Trigger Collider	Kinematic Rigidbody Trigger Collider
Static Collider		Y				
Rigidbody Collider	Y	Y	Y			
Kinematic Rigidbody Collider		Y				

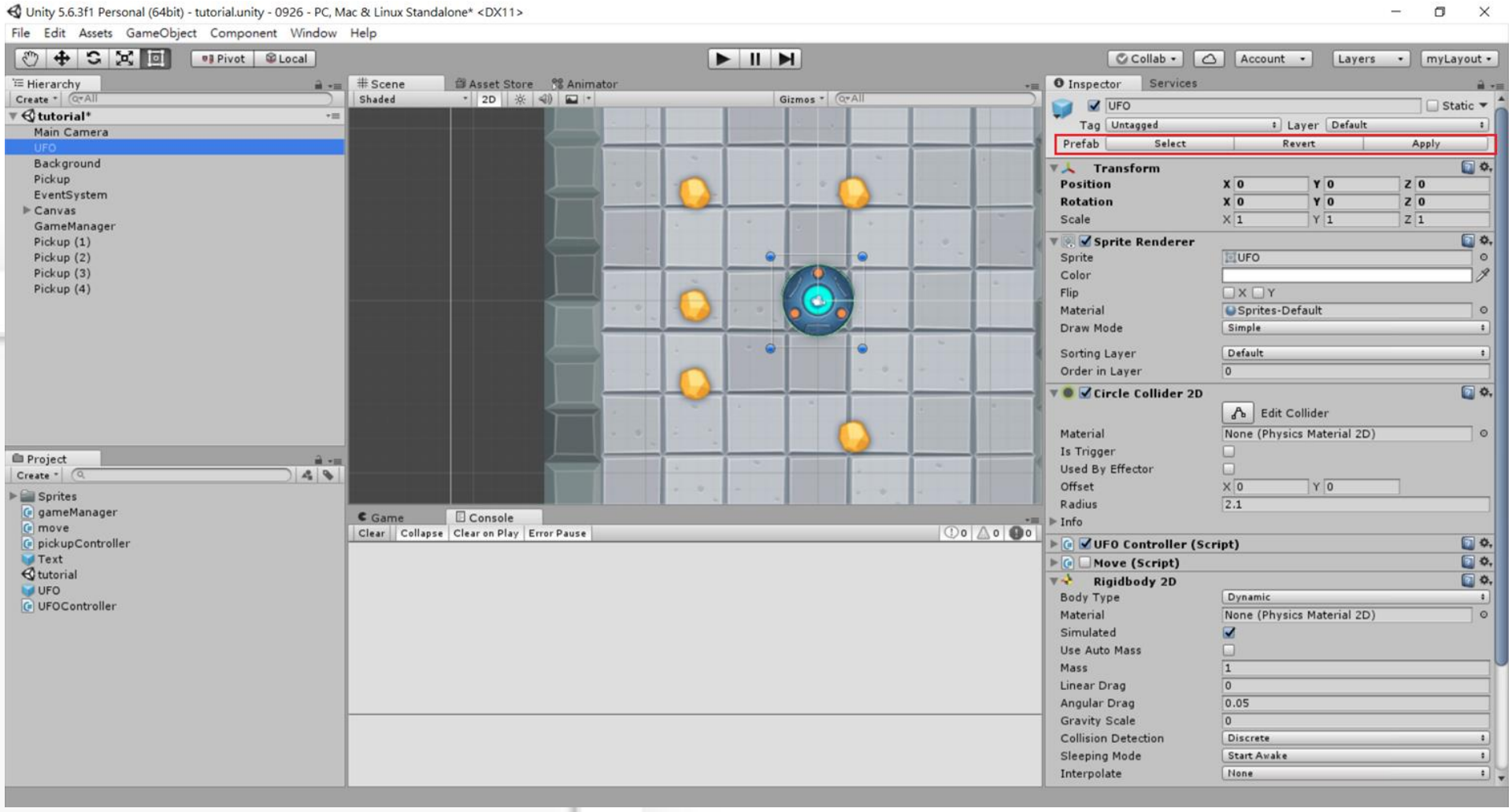
	Static Collider	Rigidbody Collider	Kinematic Rigidbody Collider	Static Trigger Collider	Rigidbody Trigger Collider	Kinematic Rigidbody Trigger Collider
Static Collider					Y	Y
Rigidbody Collider				Y	Y	Y
Kinematic Rigidbody Collider				Y	Y	Y
Static Trigger Collider		Y	Y		Y	Y
Rigidbody Trigger Collider	Y	Y	Y	Y	Y	Y
Kinematic Rigidbody Trigger Collider	Y	Y	Y	Y	Y	Y

PREFAB (預設體)

Prefab (預設體)

- 預先設定屬性的物件
- 用於生成固定性質的固定物件
- How
  - 將場景中的某一物件拖曳至Project欄位
  - 被你拖曳到project的物件會變得藍色的，代表他成為了prefab，之後你對該物件的修改，可以透過inspector欄位上方的prefab row進行設置
- 往後可以直接將帶有固定屬性的物件直接(從Project tab)拉入場景





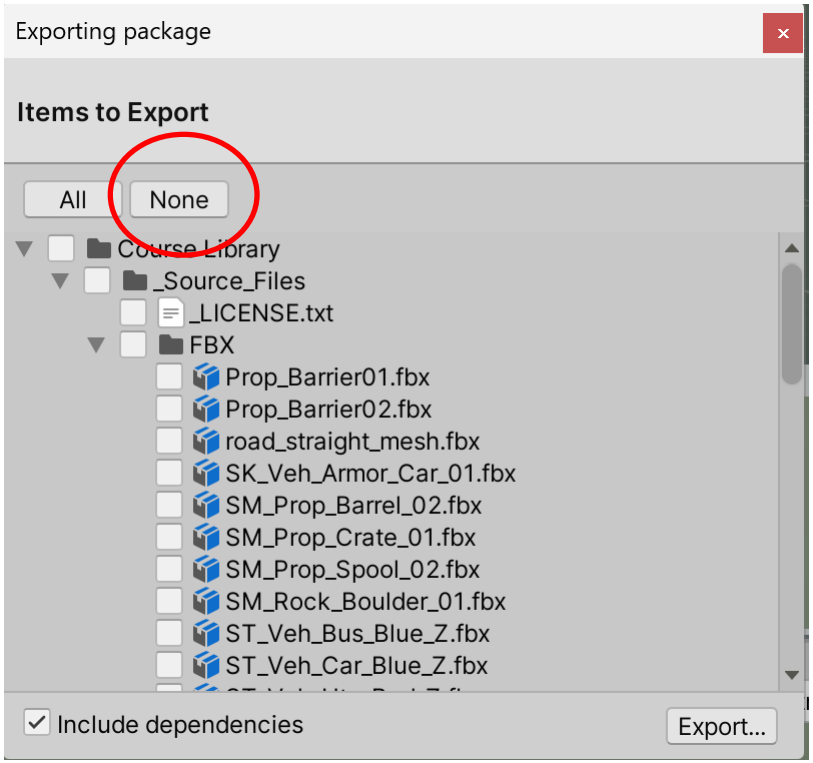
Script - the most important

- https://docs.unity3d.com/ScriptReference/
- 寫程式基本上不太可能所有method都會
- 不會的、不確定的都可以去上面查

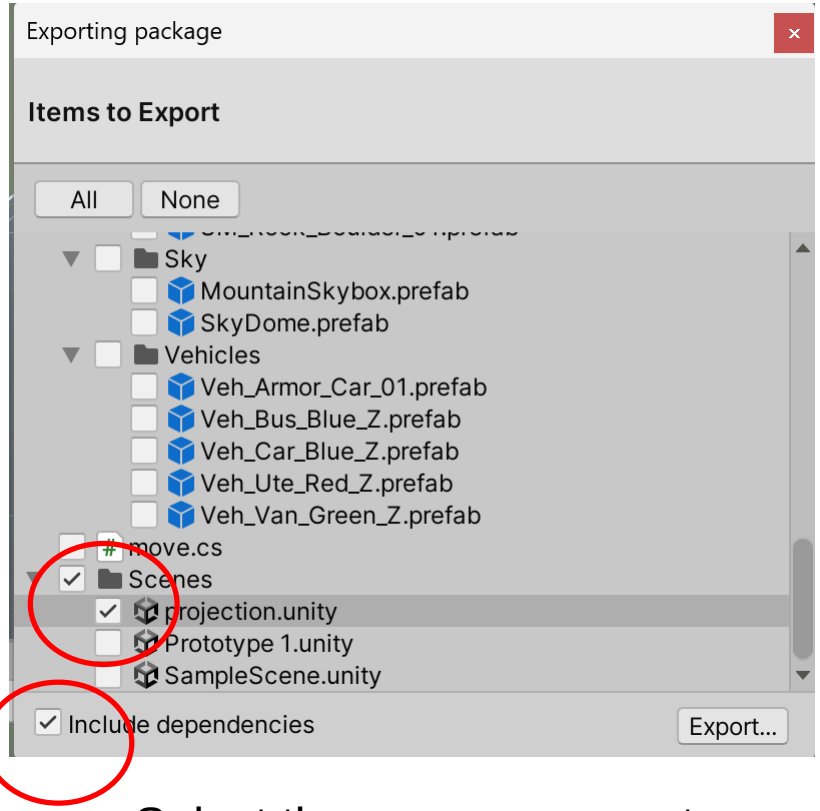
EXPORT ASSET

EXPORT scene deepened files

減少輸出檔案的大小。只選特定場景和其相關的檔案



Select none first



Select the scene you want to export

	size
Full	795K
Only one scene	5K