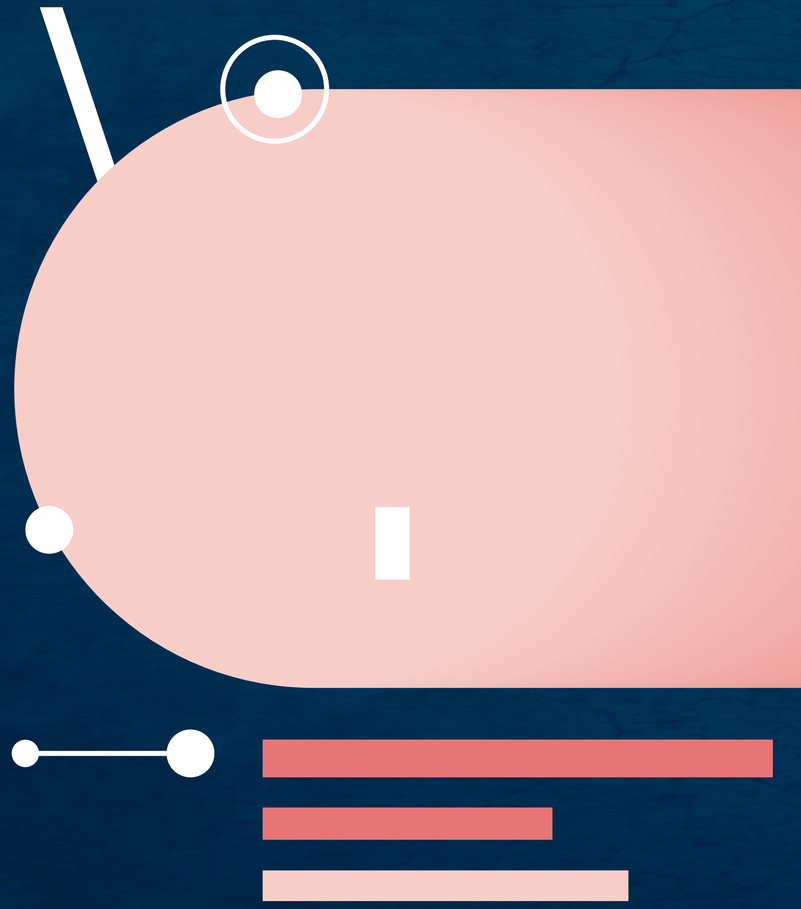
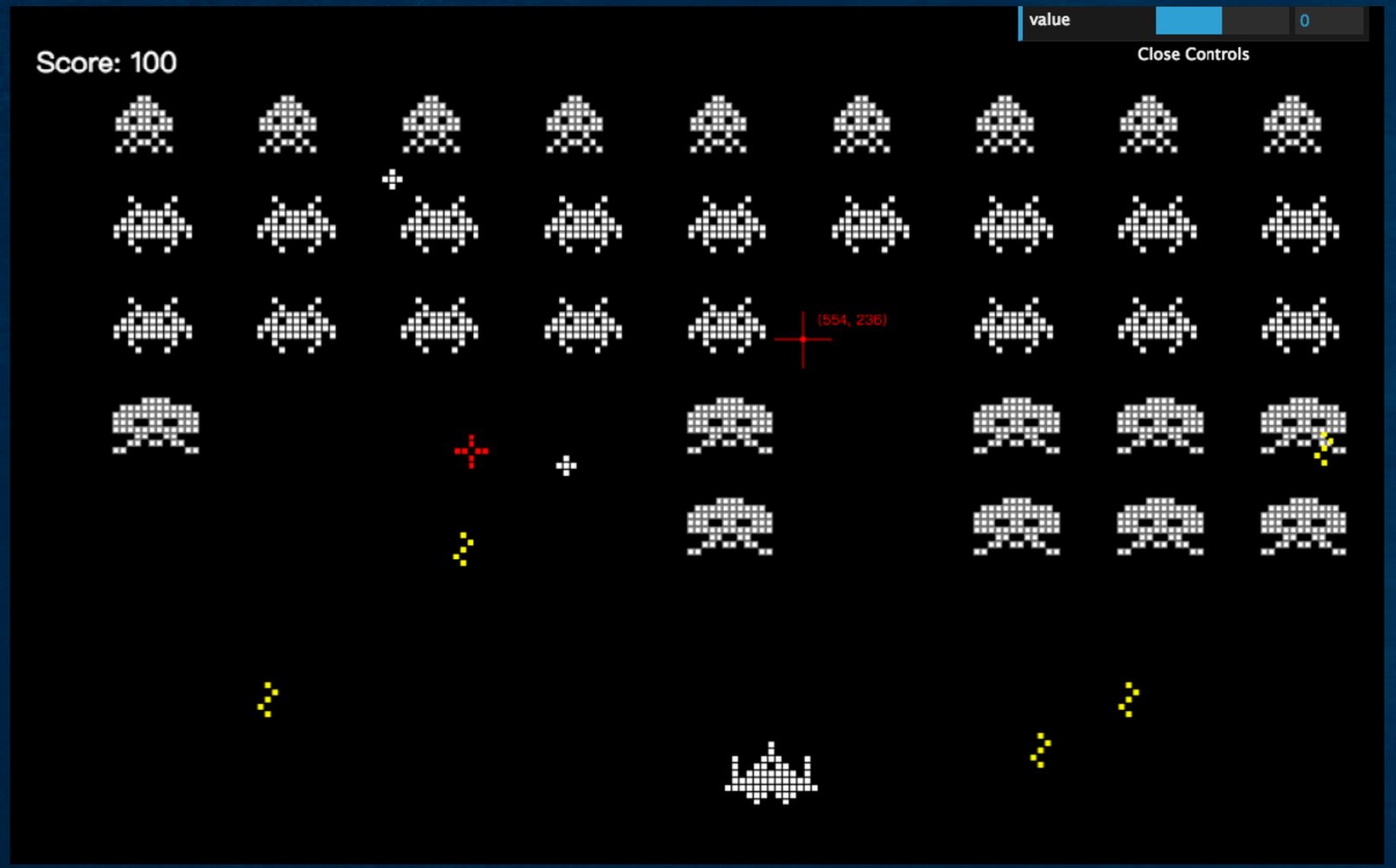
<project 12> 結合設計與繪圖製作太空侵入者

Canvas與特效動畫



太空侵入者



遊戲物件管理

```
//遊戲物件 - 所有出現在遊戲場景中的物件原型
class GameObject {
  constructor(args){↔}
  update(){↔}
  draw(){↔}
  get boundaryPoints(){↔}
  inRange(pos){↔}
  collide(obj){↔}
}
```

負責基礎的物理（位置、速度、加速度）
與偵測碰撞

```
//玩家物件，有更新大小跟有死亡控制
class Player extends GameObject{
  constructor(args){↔}
}

//敵人物件，有更新大小跟死亡
class Enemy extends GameObject{
  constructor(args){↔}
}

//子彈物件 有預設大小跟速度
class Bullet extends GameObject{
  constructor(args){↔}
}
```

遊戲物件管理

- 「場景」管理「遊戲物件 GameObject」
- 將敵人、子彈、本體、效果都視為遊戲物件放入
- 放在場景中物件會統一被渲染
- 加入時會檢查型別

```
class Scene {
  constructor(args){↔}
  init(){↔}
  //加入遊戲物件到舞台
  addChild(obj){↔}
  //移除特定物件
  removeChild(obj){↔}
  //以名稱取得物件
  getChildByName(name){↔}
  //以tag取得物件
  getChildrenByTag(tag){↔}
  update(){↔}
  draw(){↔}
}
```

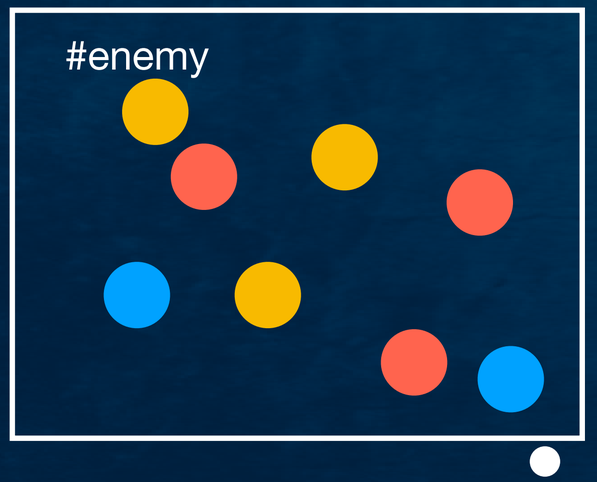
遊戲物件管理

- 要用時再根據tag/name取得物件
- 如場景選角，過濾舞台上特定種類的遊戲物件

```
let allBullets = this.getChildrenByTag("bullet")
let allEnemies = this.getChildrenByTag("enemy")
let allFlockEnemies = this.getChildrenByTag("enemy_flock")
let allEnemyBullets = this.getChildrenByTag("enemybullet")

//取得所有陣敵人
allFlockEnemies.forEach(enemy=>{
```

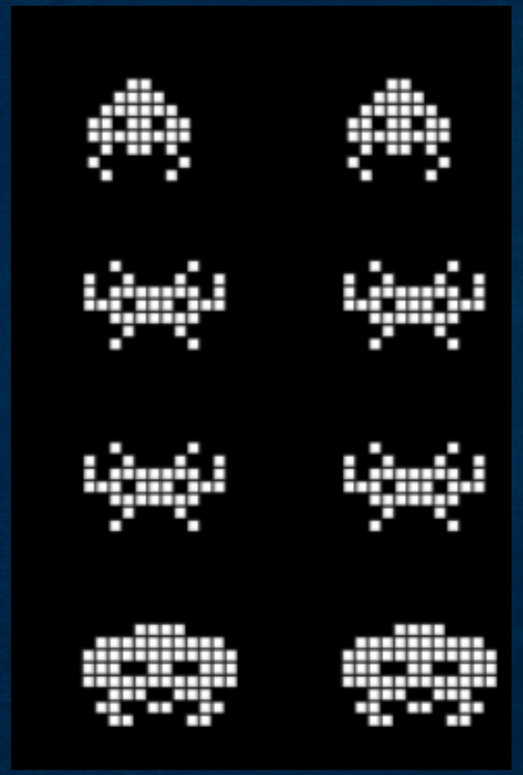
```
//加入遊戲物件到舞台
addChild(obj){
  if (obj instanceof GameObject){
    this.children.push(obj)
  }else if (obj instanceof Array){
    this.children=this.children.concat(obj)
  }else{
    console.error("Object is not a game object")
  }
}
```



取出黃色（有enemy標籤）

貼圖物件

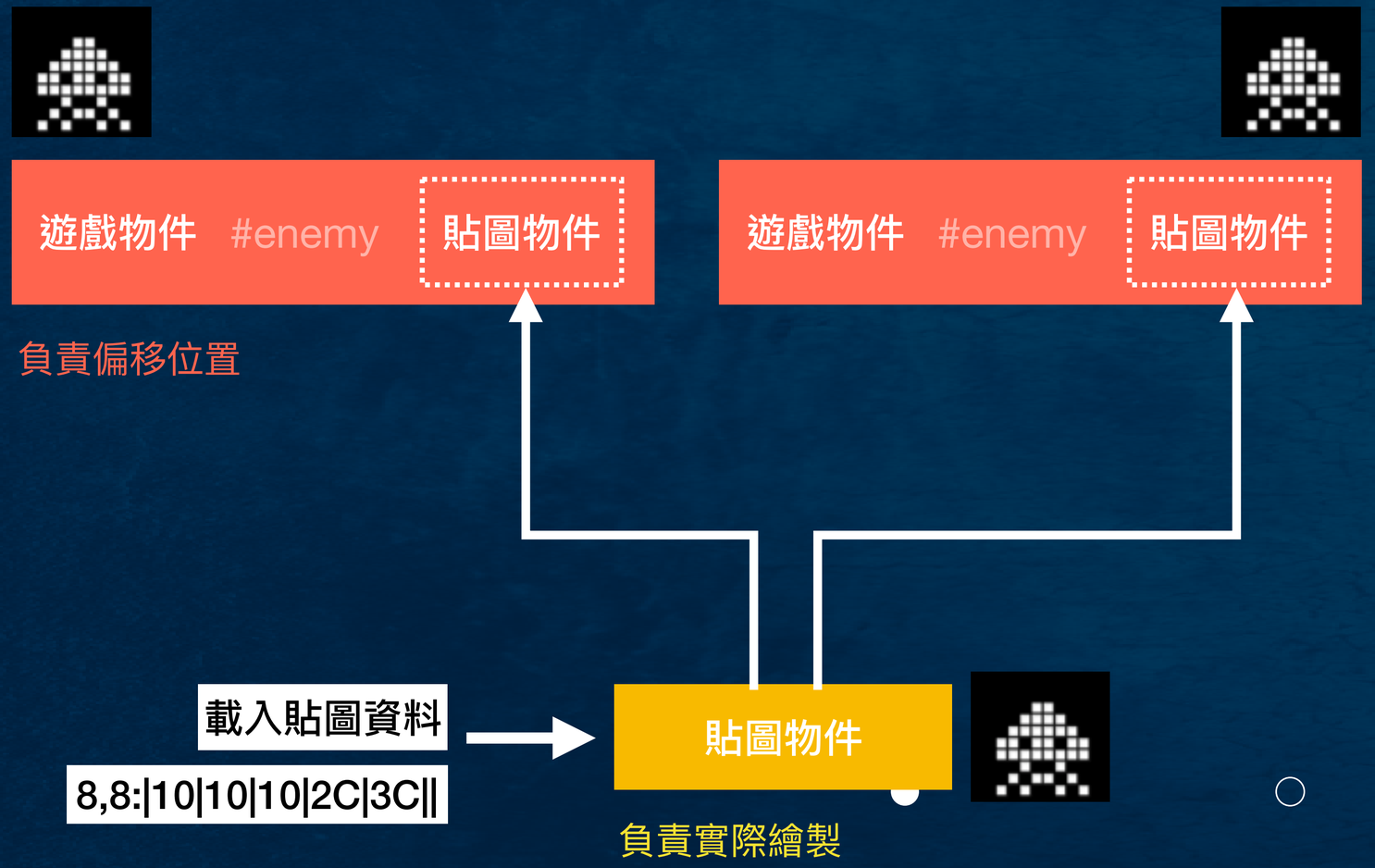
- 負責控制單一種貼圖
- 可載入文字資料
- 負責渲染圖片出來
- 可綁定在遊戲物件上



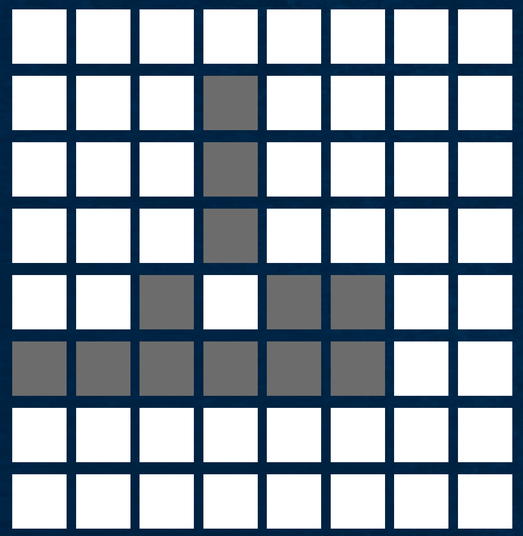
```
//貼圖物件 - 繪製點陣圖
class PixelGraphic {
  constructor(args){↔}
  //載入文字資料
  load(invaderData){↔}
  //渲染動畫
  render(){↔}
  //渲染單格影像
  renderFrame(id){↔}
  //轉換成文字
  convertBinary(text,settings){↔}
  getRealSize(){↔}
}
```

貼圖物件使用結構

- 不需要重寫每個角色的class的draw方法
- 可直接抽換貼圖物件，改變遊戲物件外觀
- 同一貼圖物件可指定給多重遊戲物件作使用



貼圖物件壓縮原理



8x8矩陣圖形

- - - - - - - -
- - - + - - - -
- - - + - - - -
- - - + - - - -
- - + - + + - -
+ + + + + + - -
- - - - - - - -
- - - - - - - -

轉換成文字表示

00000000
00010000
00010000
00010000
00101100
11111100
00000000
00000000

轉換成0與1表示

0
16
16
16
44
374
0
0

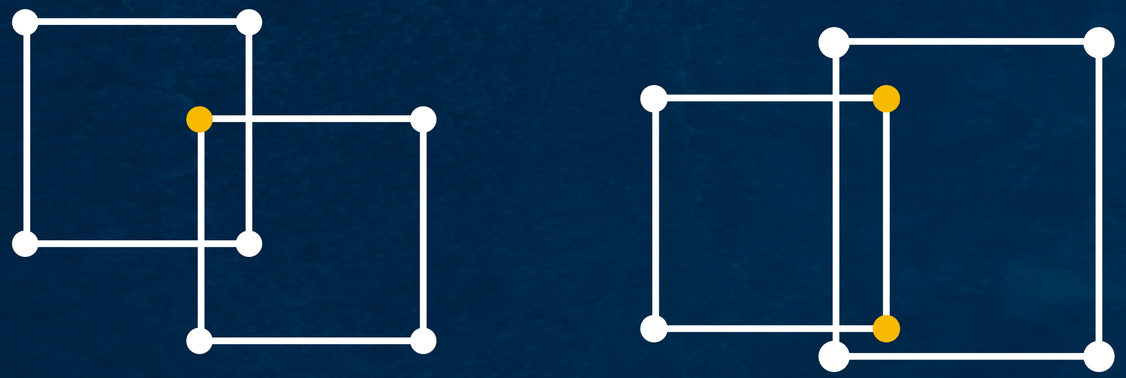
2進位換算數字

~~X~~
10
10
10
2C
FC
~~X~~
~~X~~

換算16進位

8,8:|10|10|10|2C|3C||

相撞判斷



- 有任何一個邊界點在另一方的範圍內

```
let points1 = Object.values(this.boundaryPoints)
//對方的邊界點
let points2 = Object.values(obj.boundaryPoints)
```

```
//如果自己的點出現在對方範圍內
let flag = false
points1.forEach(p=>{
  if (obj.inRange(p)){
    flag=true
  }
})

//如果對方點出現在自己範圍內
points2.forEach(p=>{
  if (this.inRange(p)){
    flag=true
  }
})
return flag
```

實作範例練習