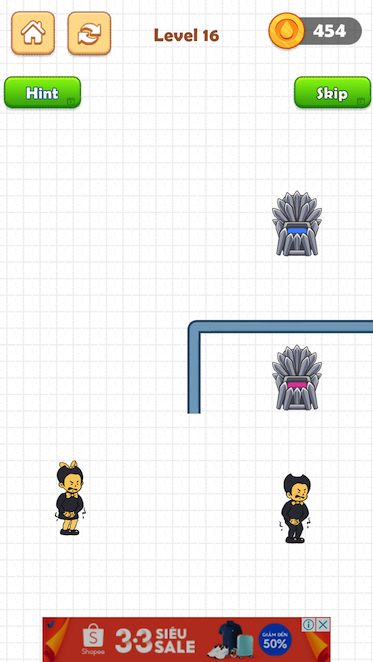
Cocos Framework Guide

1. **Các khái niệm cơ bản**
   1. **Scene:**
      * Là khung cảnh trong game
      * Một game có thể có 1 hoặc nhiều scene (Framework hiện tại sẽ chỉ sử dụng duy nhất 1 scene cho 1 game)
   2. **Screen:**
      * Là các màn hình trong game: menu\_screen, play\_screen, …
      * Scene sẽ là node cha, các screen sẽ là node con trực tiếp của scene
      * Các screen có thể chuyển qua lại lẫn nhau
      * Đặc điểm quan trọng: screen sẽ hiển thị full màn hình
      * Minh họa:

**Menu Screen**  **Play Screen**

* 1. **PopUp:**
     + Là các cửa sổ bật lên
     + Đặc điểm quan trọng: chỉ hiển thị 1 phần màn hình, đè lên nội dung của screen (không hiển thị full màn hình như screen)
     + Minh họa:



Popup Setting

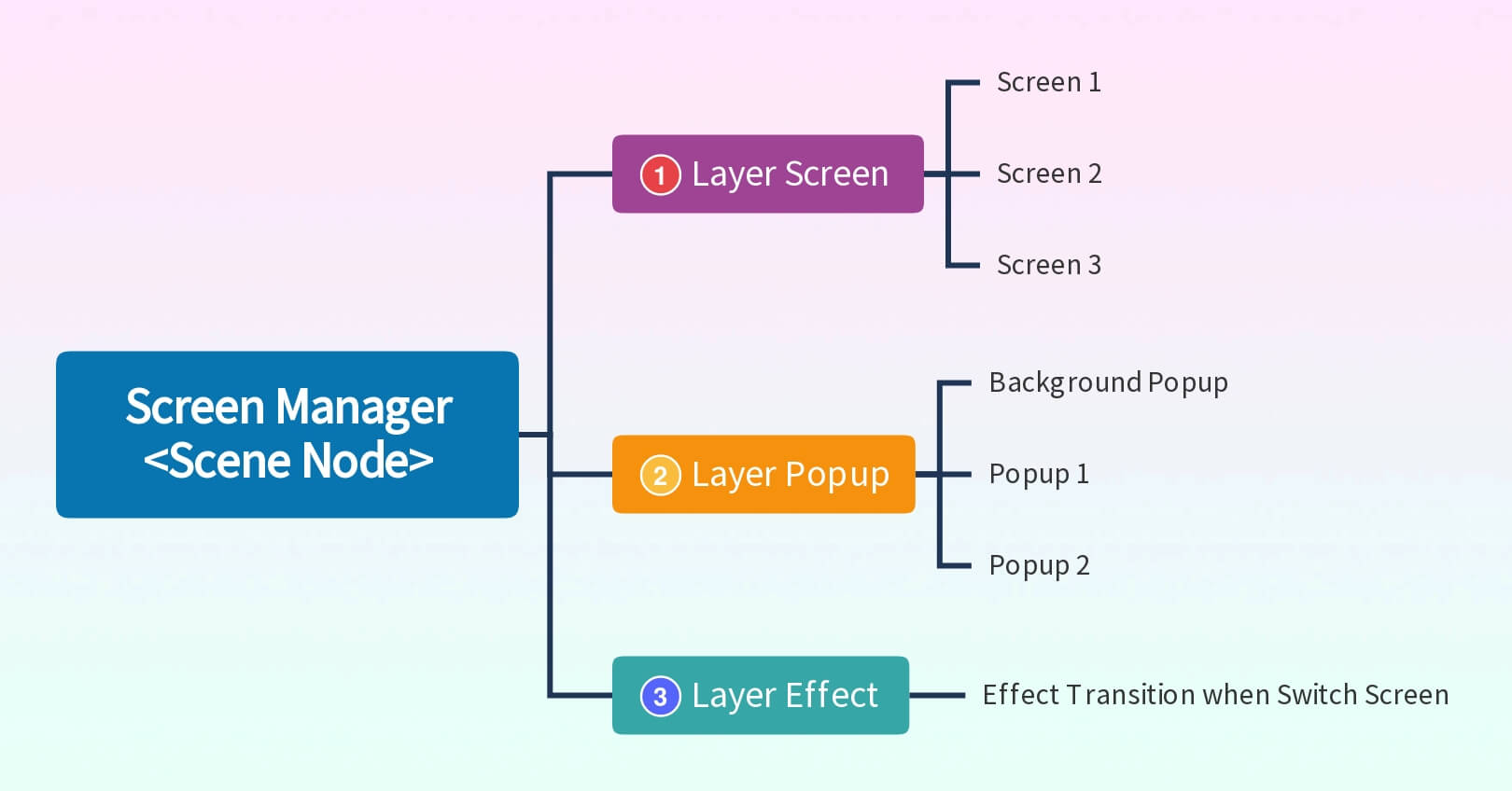
* Câu hỏi: Vậy thì khi nào thì chúng ta thiết kế là screen, khi nào là popup?
  + Khi cần thiết kế 1 màn hình fullscreen => chọn **Screen**
  + Khi cần thiết kế 1 bảng nhỏ ko fullscreen => chọn **Popup**

1. **Logic quản lý các scene, screen (ScreenManager)**
   1. Như đã nói phía trên, Framework sẽ thiết kế mỗi game chỉ có 1 **Scene**.
   2. Scene sẽ có 1 component quản lý các screen, popup đó là: **ScreenManager**
   3. Trong **ScreenManager** sẽ có các Node con đảm nhận chức năng riêng biệt như:
      * Layer Screen (Node): là node cha để add các screen vào
      * Layer Popup: là node cha để add các **PopUp**
      * Layer Effect: node cha để add các effect khi chuyển cảnh giữa các screen
      * …

Các layer trên cũng được sắp xếp theo thứ tự hiển thị sau:

**Screen** < **Popup** < **Effect** <…

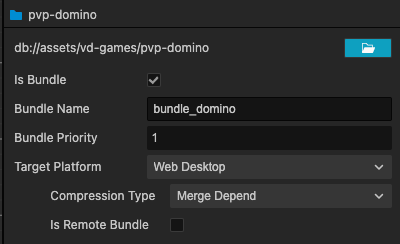
Minh họa:



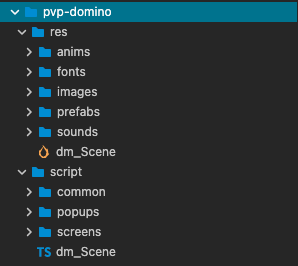
1. **Các bước cụ thể để tạo 1 project game**
   1. Khi nhận được Project Framework (từ người quản lý), chúng ta sẽ nhìn thấy như sau:



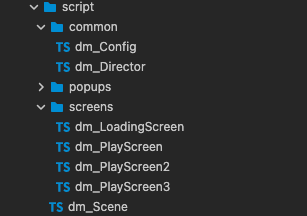
* + - * + Chỉ quan tâm tới thư mục “vd-games”, trong đó sẽ chứa từng game riêng lẻ
  1. Tạo thư mục game của mình, chú ý cú pháp đặt tên “pvp-tên game”
  2. Set thư mục game của mình thành 1 bundle:
     + Nhìn sang bên phía Inspector: click isBundle
     + Đặt tên bundle, cú pháp như sau: “bundle\_tên game”



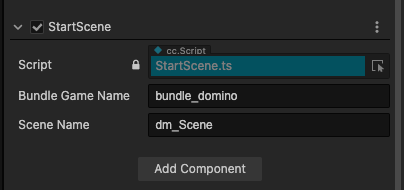
4. Tạo các thư mục script, res và quản lý tài nguyên theo thư mục để dễ quản lý.



* + Recommend (Không bắt buộc): Thư mục nên tổ chức giống như mẫu trên để có sự thống nhất giữa các project.
  + Đặt tên file script (Bắt buộc): cú pháp sẽ là **“tiền tố” + “\_” + “script name”.** Trong đó tiền tố sẽ tùy theo game, ví dụ ở game Domino phía trên thì tiền tố là “dm”



1. **Run Game**
   1. **StartScene** (Path: vd-boot/StartScene)
      * Đây là Scene bắt đầu của game
      * Trong Canvas của **StartScene** có gắn script “StartScene.ts”
      * Truyền 2 tham số: Bundle Game Name và Scene Name để chỉ định scene nào tiếp theo sẽ được chạy.



* 1. **Scene Game**
     + Trong ví dụ trên thì Scene Game là dm\_Scene.
     + Canvas của dm\_Scene có 1 script là dm\_Scene.ts

Script này sẽ load prefab loading\_screen và show screen lên nhờ hàm:

VDScreenManager.instance.pushScreen(…);

* 1. **Loading Screen**
* Có script dm\_LoadingScreen.ts để load tài nguyên (đọc script trong thư mục game mẫu pvp-domino để hiểu logic load)
* Sau khi load xong thực hiện chuyển màn sang các screen tiếp theo (tùy kịch bản game)

1. **Các đối tượng và hàm cơ bản trong Framework**
   1. **Screen**
      * + Script: VDScreenManager
        + Các tính năng cơ bản: show screen, switch screen (có transition hoặc không)
        + Chú ý: script nào điều khiển screen thì cần extend **VDBaseScreen**

(xem ví dụ ở file dm\_PlayScreen.ts )

|  |  |  |
| --- | --- | --- |
| Tên hàm | Tham số | Cách dùng |
| initWithRootScreen | - screen: cc.Prefab: screen cần hiện  - callback: trả về screen | * + - Show screen đầu tiên trong scene |
| pushScreen | screen2: cc.Prefab, callback,  animated: (có hiệu ứng transition  hay không) | * + - Show screen2 lên     - Screen1: bị tách khỏi node cha,   vẫn tồn tại để có thể back về |
| popScreen | animated | * + - Back về screen gần nhất |
| popToScreenAtIndex | screenIdx, animated | * + - Back về 1 screen với id truyền vào |
| popToRootScreen | animated | * + - Back về screen đầu tiên |
| replaceScreenAtIndex | screen, screenIdx, callback | * + - Replace screen hiện tại bằng   Screen mới, screen hiện tại sẽ bị  Destroy, ko thể back về được nữa |

* + - * Ví dụ: VDScreenManager.instance.pushScreen(prefab, (screen) => {});

## **Popup.**

* + - * Các tính năng:

+ show Popup: dùng VDScreenManager

+ hidePopup: có thể tự tắt bằng hàm hide của VDPopup hoặc dùng VDScreenManager.removePopup

* + - * Chú ý: script nào điều khiển popup thì cần extend **VDBasePopup**

(xem ví dụ ở file dm\_Popup1.ts )

|  |  |  |
| --- | --- | --- |
| Tên hàm | Tham số | Cách dùng |
| showPopupFromPrefabName  (VDScreenManager.instance) | - resPath: đường dẫn tới Prefab  - callback:  - hideWhenTouchOnBackground:  ẩn Popup khi touch vào background  - animated: có hiệu ứng hay không  - showBackgroundLayer: có hiển thị  background hay không | * + - Show popup dựa vào   đường dẫn tới prefab |
| showPopupFromPrefab  (VDScreenManager.instance) | -prefab: truyền vào bằng Prefab | * + - Tương tự như trên nhưng   truyền vào Prefab |
| hide  (this.hide() – Popup đó phải  Extend VDBasePopup) |  | * + - Đóng Popup |

* 1. **Audio**
     + - Chúng ta sẽ có 3 loại:
     + **Background Music (BGM):** âm thanh nền, lặp đi lặp lại
     + **Clip**: âm thanh dài khoảng 10-20s, khi phát lên thì tạm thời pause BGM lại để chạy âm thanh đó, chạy xong có thể resume lại BGM
     + **Effect**: âm thanh ngắn (ting, gun). Đặc điểm của âm thanh này là có thể phát nhiều âm cùng lúc, trong khi BGM với Clip thì chỉ phát 1 âm trong 1 thời điểm.
       - Function:
     + BGM:
       - VDAudioManager.instance.playBGM (BGM\_name)
       - VDAudioManager.instance.isMutingMusic = true/ false (pause/resume)
     + Clip:
       - VDAudioManager.instance.playClip(clip\_name, loop, resumeBGM, callback);
       - VDAudioManager.instance.stopClip();
     + Effect:
       - VDAudioManager.instance.playEffect(effect\_name, loop, resumeBGM, callback);
       - VDAudioManager.instance.stopEffectByName(effect\_name, fade): fade là âm thanh giảm dần âm lượng -> 0
       - Lưu ý: Tạm thời chúng ta sẽ để âm thanh vào 2 thư mục như sau:
     + BGM, Clip: res/sounds/bgm/
     + Effect: res/sounds/sfx/

(Framework sẽ update để linh hoạt chỗ này sau)

**Lời kết:**

* + - Đặc biệt chú ý những phần bôi đỏ
    - Các bạn có thể đọc và run project game mẫu để hiểu thêm cách dùng Framework (vd-games/pvp-domino)
    - Framework còn nhiều hạn chế và bug, hi vọng mọi người sẽ góp ý để mình cải thiện framework hơn nữa. Many thanks!

Table of Contents

[Cocos Framework Guide 1](#_Toc128499655)

[**I.** **Các khái niệm cơ bản** 1](#_Toc128499656)

[**1.** **Scene:** 1](#_Toc128499657)

[**2.** **Screen:** 1](#_Toc128499658)

[**3.** **PopUp:** 2](#_Toc128499659)

[**II.** **Logic quản lý các scene, screen (ScreenManager)** 2](#_Toc128499660)

[**III.** **Các bước cụ thể để tạo 1 project game** 3](#_Toc128499661)

[**IV.** **Run Game** 4](#_Toc128499662)

[1. **StartScene** (Path: vd-boot/StartScene) 4](#_Toc128499663)

[**2.** **Scene Game** 5](#_Toc128499664)

[**3.** **Loading Screen** 5](#_Toc128499665)

[**V.** **Các đối tượng và hàm cơ bản trong Framework** 5](#_Toc128499666)

[**1.** **Screen** 5](#_Toc128499667)

[**2.** **Popup.** 5](#_Toc128499668)

[**3.** **Audio** 6](#_Toc128499669)