Bài 8: Tối ưu game và các phương pháp tăng lượt download trên Google Play

Giảng viên:

Mục tiêu

- Tối ưu game
- Phương pháp cải thiện số lượt download

Tối ưu hoá game trong unity

- **Định nghĩa tối ưu hoá:**
- Tối ưu hoá là một phần trong quá trình phát triển game ,nói một cách đơn giản nó là quy trình để cải tiến tính hiệu quả của hiệu năng game của bạn. Vấn đề tối ưu hoá này đặc biệt quan trọng trong quá trình phát triển game trên mobile .
- Mỗi nền tảng phát triển game đều có những hạn chế về năng lực xử lý của CPU(bộ xử lý trung tâm) và GPU (bộ xử lý đồ hoạ). Đặc biệt trong các cảnh game 3D với độ chi tiết cao thì nhu cầu xử lý sẽ càng cao . Khi nhu cầu xử lý này trở nên quá tải thì hiệu năng game sẽ giảm và gây cảm giác khó chịu cho người dùng như là load game lâu, game chạy chậm và bị giật hoặc nghiêm trọng hơn thì có thể bị treo hệ thống .
- Quy trình tối ưu hoá là chuỗi các quyết định hay thoả hiệp giữa việc duy trì hiệu năng game và chất lượng đồ hoạ .

Tối ưu hoá game trong unity

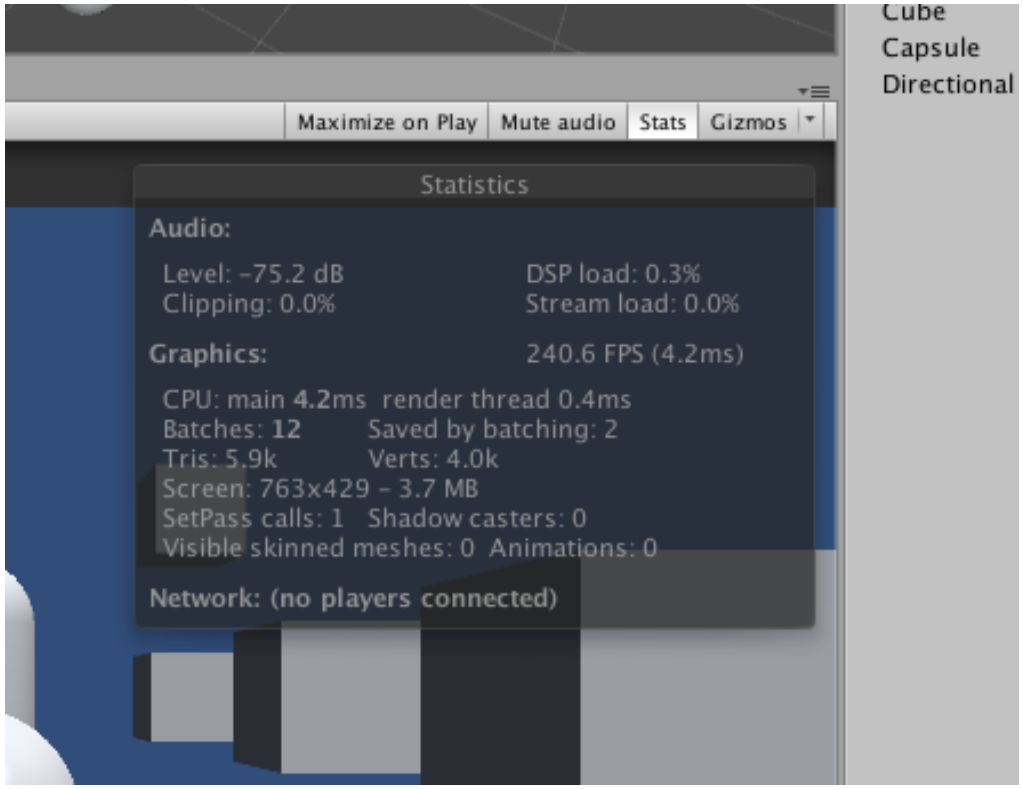
- **Hiệu năng hướng tới:**
- Hiệu năng hướng tới thường được biểu thị theo tốc độ frame mong muốn hay số frame trên giây (fps).
- Nếu tốc độ frame thấp thì người chơi mất cảm giác chuyển động.
- Tốc độ frame cao sẽ mang lại trải nghiệm trực quan tốt hơn nhưng trong các game phức tạp đòi hỏi nhiều xử lý trên frame vẫn có thể chạy trơn chu trong một khoảng tốc độ frame vừa phải nhưng vẫn chấp nhận được .

Tối ưu hoá game trong unity

- **Hiệu năng hướng tới:**
- Việc xác định hiệu năng hướng tới đầu tiên bạn nên biết rõ game của mình chạy trên nền tảng nào.
- Nếu game chạy trên các máy console và PC có bộ xử lý mạnh mẽ thì có thể có thể ta không cần quan tâm lắm nhưng trên nền tảng di động thì chúng ta bắt buộc phải quyết định sớm trong quá trình thiết kế game để hiệu năng game tối ưu nhất có thể.
- Đặc điểm hạn chế của các thiết bị di động là cấu hình thấp cả về CPU và GPU, hơn nữa trên thị trường có rất nhiều loại thiết bị khác nhau với cấu hình cũng không hề giống nhau vì vậy việc để game của ta có thể chạy tốt ở tất cả các máy là một vấn đề lớn.
- Chúng ta phải điều tra các thiết bị phổ biến để xác định một tốc độ frame hợp lý để game chạy tốt trên nhiều máy nhất có thể .

Tối ưu hoá game trong unity

- **Theo dõi mục tiêu:**
- Trong trình soạn thảo Unity bạn hãy chạy một cảnh bất kỳ. Trong menu trên cùng của khung nhìn Game bạn hãy nhấn nút Stats, bạn sẽ thấy một bảng các thông số thống kê hiệu năng hiện ra gồm cả tốc độ frame trên giây (FPS) .



Tối ưu hoá game trong unity

- **Theo dõi mục tiêu:**
- Chỉ số Batches thậm chí còn chỉ ra thông tin chi tiết hơn đó là số lượng vật mà CPU cần render.
- Việc hiểu rằng một đối tượng game có thể được render nhiều lần trên frame đối với mỗi hiệu ứng áp dụng trên đối tượng này là rất quan trọng, chẳng hạn như ánh sáng ,đổ bóng và phản chiếu điểm ảnh.
- Ta nên dùng ít các hiệu ứng như vậy hơn bằng cách chỉ áp dụng khi chúng có đóng góp đáng kể cho giao diện game sẽ giảm bớt các lời gọi hàm vẽ draw call, nghĩa là giảm bớt công việc cho CPU.

Tối ưu hoá game trong unity

- **Theo dõi mục tiêu:**
- Ta có thể thiết lập tốc độ frame nhắm đến bằng cách sử dụng một script đơn giản.

```
public int frameRate = 50;
void Awake()
{
    Application.targetFrameRate = frameRate;
}
```

- Sau đó áp dụng script này vào scene chúng ta cần thiết lập tốc độ frame.

Tối ưu hoá game trong unity

- **Theo dõi mục tiêu:**
- Chú ý rằng tốc độ frameRate = 50 thậm chí có thể không có tác dụng gì cả đó là bởi vì tốc độ frame nhắm tới không phải là một giới hạn tuyệt đối.
- Nếu muốn xem sự khác nhau đáng chú ý ta hãy chỉnh frameRate = 25; sau đó chạy lại bạn sẽ thấy tốc độ frame bây giờ sẽ trong khoảng gần 25 hơn. Khi bạn chỉnh về -1 thì đây là giá trị mặc định, giá trị này thiết lập game về tốc độ 50-60 fps và báo cho các thiết bị render nhanh nhất có thể .

Tối ưu hoá game trong unity

- **Thời điểm nào và nơi nào cần tối ưu hoá?**
- Các quyết định tối ưu hoá được thực hiện trong giai đoạn thiết kế phần lớn dựa trên những giới hạn đã biết về nền tảng nhắm đến.
- Trong quá trình phát triển, việc tuân theo các khuyến nghị về kinh nghiệm thực tiễn tốt nhất sẽ tránh việc hiệu năng bị suy giảm bởi những kỹ thuật phát triển kém hiệu quả.
- Ngoài ra quá trình tối ưu hoá khá động liên quan tới những đặc trưng riêng của game khi các chi tiết nhiều vô số, điều này có thể tác động tới hiệu năng game.

Tối ưu hoá game trong unity

- **Giai đoạn thiết kế?**
- Giai đoạn thiết kế sẽ có nhiều quyết định hơn là chỉ chọn tốc độ frame nhắm đến.
- Khi xem xét đồ hoạ game khả năng của nên tảng nhắm đến sẽ được xem xét.
- Kinh nghiệm chung là không sử dụng thêm bất kỳ đỉnh nào cho một mô hình nhiều hơn số lượng cần thiết, tuy vậy giới hạn render của CPU phụ thuộc vào mọi thứ trong cảnh, gồm cả các yếu tố khác như việc sử dụng màu sắc, nguồn sáng và đổ bóng chứ không chỉ một mô hình hay một yếu tố nào khác.
- Bạn nên tham khảo các tài liệu hướng dẫn của nền tảng nhắm đến đối với các trường hợp cụ thể .

Tối ưu hoá game trong unity

- **Giai đoạn thiết kế?**
- Các thiết bị di động không thể xử lý nhiều hơn 100 000 đỉnh còn PC thì có thể xử lý nhiều hơn tới vài triệu đỉnh.
- Với game di động bạn có thể chọn một mức chi tiết nền thấp hơn nhiều mứa so với khi bạn dùng trên game PC để bảo toàn các chi tiết của mô hình đối tượng game chính .
- Một giới hạn khác nữa của thiết bị di động là kích thước file tải về . Việc kết hợp tái sử dụng các mô hình, texture và chất liệu vào thiết kế game sẽ giảm kích thước file chung.

Tối ưu hoá game trong unity

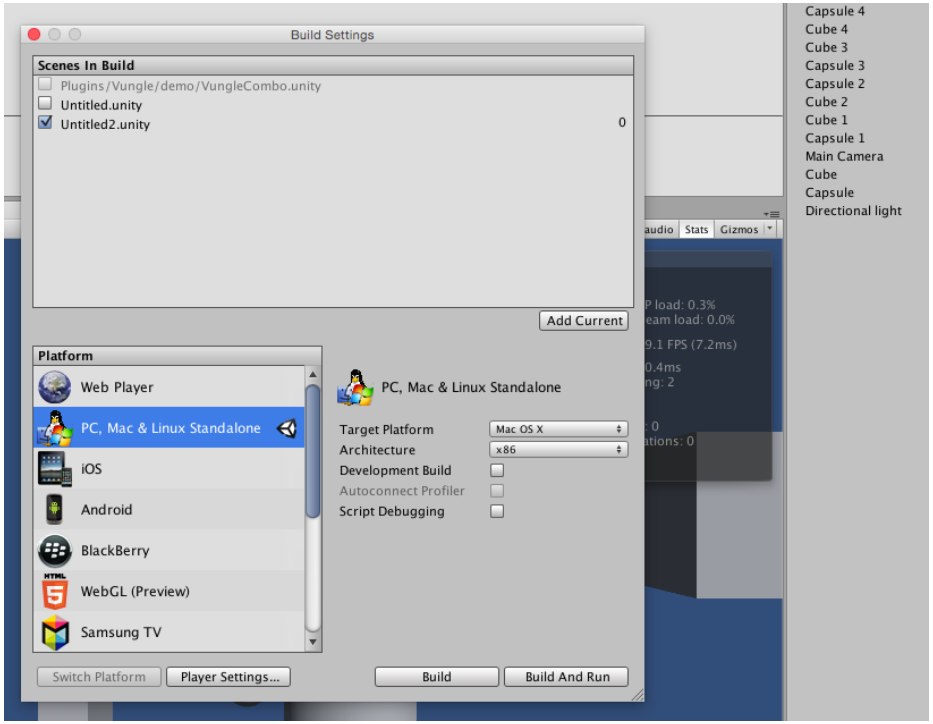
- **Giai đoạn phát triển?**
- Mục đích tối ưu hoá là để duy trì mức độ hiệu năng game mà bạn nhắm đến. Khi game trình diễn với mức độ hiệu năng đó từ lúc bắt đầu cho tới lúc hoàn tất thì công việc của bạn hoàn tất .
- Tác động trực quan trên người chơi đóng góp đáng kể đến trải nghiệm game, đó là lý do vì sao các công cụ đồ hoạ game đi chi tiết tới mức điểm ảnh và yêu cầu nhiều lần render để kết hợp mọi chi tiết cuối cùng của nguồn sáng, phản chiếu, đổ bóng ...
- Đồ hoạ trực quan không chỉ là tác nhân duy nhất có thể cản trở hiệu năng game. Vật lý, animation, âm thanh, và nhiều thứ khác cũng có thể khiến cho hiệu năng game bị ảnh hưởng.

Tối ưu hoá game trong unity

- **Giai đoạn phát triển?**
- Khi bạn xây dựng game và test lại nhiều lần, sự tắc nghẽn có thể xuất hiện ở một số điều kiện đáng chú ý của gameplay , chẳng hạn khi một mô hình phức tạp xuất hiện trong khung hình, trong một chuỗi animation cụ thể, hay kèm theo hệ thống hiệu ứng hạt. Tốt nhất chúng ta nên trực tiếp giải quyết các vấn đề đó .
- Nếu nguyên nhân tắc nghẽn không rõ ràng, Unity có nhiều công cụ giúp bạn xác định. *Editor Profiler* là một công cụ chuyên dụng để đánh giá game cả trên Editor hay trên thiết bị kiểm thử, tiếc là công cụ này chỉ có trên bản Unity Pro. Các công cụ profiler dựng sẵn đã có dành cho cả iOS và android . Tuy vậy bạn vẫn có thể có được các thông tin hữu ích về game của mình từ Build log .

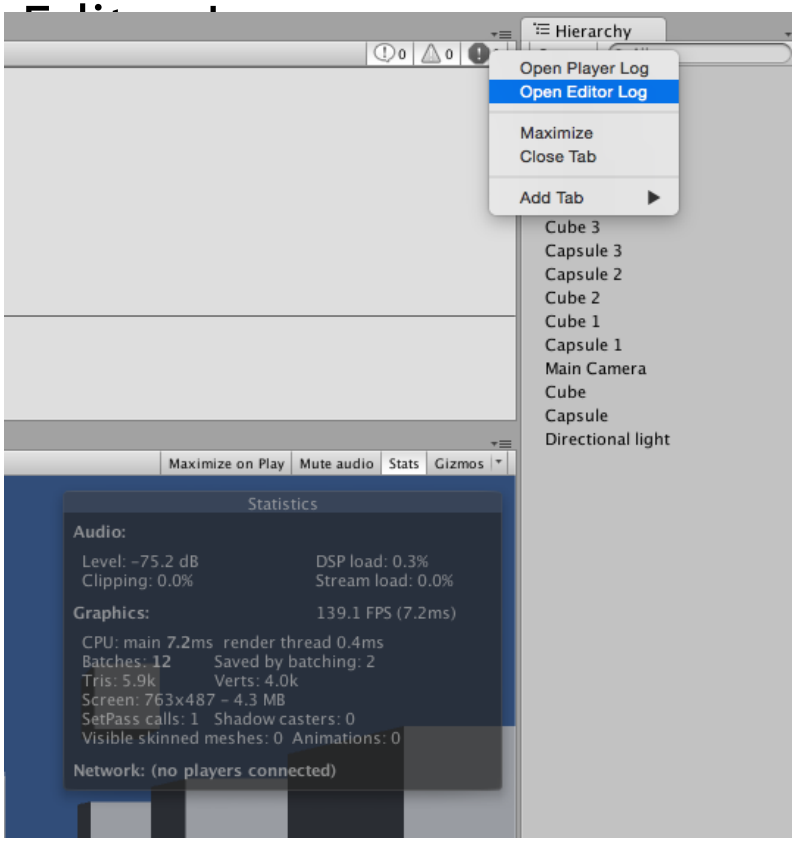
Tối ưu hoá game trong unity

- **Cách tạo build log?**
- Trong menu trên cùng trình soạn thảo Unity chọn **File-> build setting**. Hãy chắc chắn chọn tuỳ chọn PC, Mac & Linux Stangalone trong cửa sổ plasform để Unity build được kiểu file phù hợp cho nền tảng bạn nhắm đến. Sau đó nhấn build rồi chọn nơi lưu file rồi nhấn save.



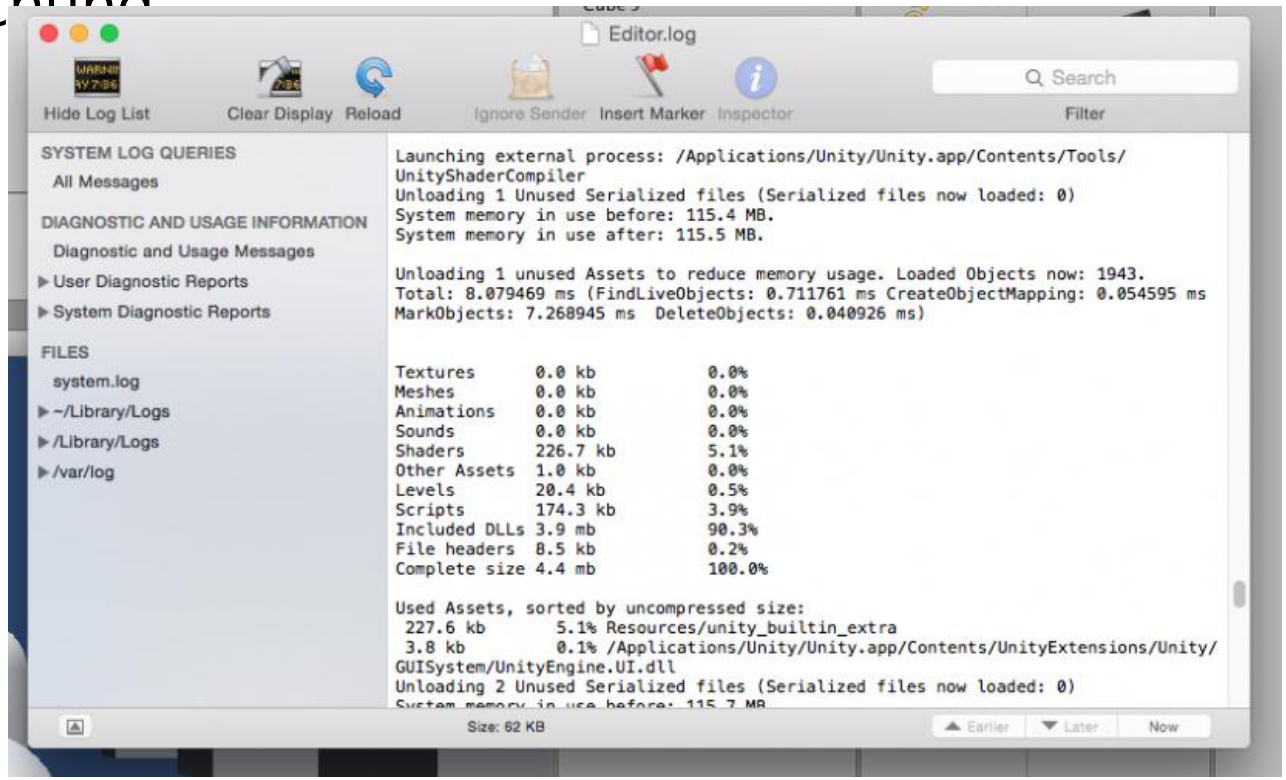
Tối ưu hoá game trong unity

- **Cách tạo build log?**
- Sau khi build hoàn tất bạn có thể truy cập build log bằng cách chọn tab Console trong bảng Project, tìm biểu tượng menu thả xuống ở góc trên bên phải rồi chọn Open



Tối ưu hoá game trong unity

- **Cách tạo build log?**
- Build log sẽ hiện ra trong cửa sổ Console. Cùng với các thông tin khác, bạn có thể thấy danh sách các kiểu tài nguyên tạo nên game cùng với kích thước tương đối của chúng



Tối ưu hoá game trong unity

- **Cách tạo build log?**
- Ở ngay bên dưới bạn sẽ thấy danh sách Used Assets (tài nguyên đã dùng). Thời gian tải game bị ảnh hưởng bởi kích thước này, còn người chơi game thì muốn được chơi game ngay.
- Vậy ta nên tối ưu hoá ở đây bằng cách bỏ đi các tài nguyên không dùng đến trong dự án khi phát triển. Người tải game chú ý tới kích thước file tải về và họ không hề muốn tải về một file nặng, hơn nữa nhà cung cấp dịch vụ cũng giới hạn kích thước file tải về.
- Vì vậy ta nên giảm sao cho kích thước file tải về càng nhỏ càng tốt. Bằng cách so sánh danh sách Used Assets với nội dung thư mục Assets bạn có thể xoá đi một cách an toàn các file không dùng đến .

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
  - Đặt biến kiểu tĩnh và chỉ thị #pragma strict _
  - Lưu đệm các tìm kiếm component
  - Sử dụng trigger
  - Object pooling
  - Coroutine

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
- Đặt biến kiểu tĩnh và chỉ thị #pragma strict _:
  - Trình soạn thảo MonoDevelop tạo ra script cho bạn và luôn bắt đầu với dòng #pragma strict. Dòng này báo cho trình biên dịch biết để dịch đoạn mã theo sau "một cách chặt chẽ" (strictly), tức là bắt buộc bạn dùng kiểu tĩnh nếu không sẽ có lỗi biên dịch được ném ra.
  - Đặt kiểu tĩnh ám chỉ việc đặt kiểu biến khi bạn khai báo chúng. Trong dòng mã sau bạn khai báo tường minh biến myValue thuộc kiểu int

```
var myValue : int = 2;
```

```
var myValue = 2;
```

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
- Đặt biến kiểu tĩnh và chỉ thị #pragma strict _:
  - Unity sẽ tự động chuyển biến này thành mã kiểu tĩnh sử dụng phương pháp có tên là suy diễn kiểu (type inference). Mặc dù khả năng này cho phép viết mã đơn giản hơn nhưng nếu biến không thể suy diễn kiểu được thì Unity sẽ phụ thuộc vào việc định kiểu động .
  - Trong định kiểu động (dynamic typing), Unity phải chỉ ra kiểu biến nào dựa trên giá trị được gán cho biến. Việc "chỉ ra" này sẽ làm tốn thời gian và do đó ảnh hưởng đến hiệu năng game. Việc dùng định kiểu tĩnh sẽ giúp cho hiệu năng game ổn định, còn chỉ thị #pragma strict là để đảm bảo bạn làm điều này.

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
- Lưu đệm các tìm kiếm component:
  - Kỹ thuật viết script này thường hiệu quả với các script được dùng thường xuyên, nhưng đổi lại đòi hỏi viết nhiều mã hơn một chút.
  - Hàm GetComponent() là một ví dụ về tìm kiếm (lookup).
  - Việc tìm component trong đối tượng game sẽ tiêu tốn thời gian và ảnh hưởng tới hiệu năng.
  - Ý tưởng ở đây là tìm tham chiếu một lần duy nhất rồi lưu đệm hoặc lưu tham chiếu trong một biến private để sẵn sàng sử dụng trong các script về sau.
  - Nói cách khác hãy tránh dùng hàm GetComponent() trong hàm Update() hoặc hàm FixedUpdate() bất cứ nơi nào có thể

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
- Sử dụng trigger:
  - Giảm và tránh các tính toán theo từng frame đều sẽ giúp cải thiện hiệu năng
  - Ví dụ để xoá một viên đạn đi bằng cách theo dõi hàm Destroy() sau 2 giây bạn viết mã như sau :

```
Destroy(bullet, 2);
```

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
- Sử dụng trigger:
  - Mặc dù đây là sự cải tiến để tránh việc có quá nhiều viên đạn trong chương trình , nhưng Unity vẫn phải theo dõi mỗi đối tượng viên đạn và kiểm tra vòng đời của chúng cho tới khi nào hết 2 giây. Một giải pháp khác là hàm Destroy() sẽ sử dụng khi viên đạn chạm vào một vùng trigger nào đó mà ta chỉ định thay vì việc phải kiểm tra liên tục

```
void OnTriggerEnter(Collider other)
{
    Destroy(bullet);
}
```

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
- Object pooling:
  - Object pooling lấy việc tối ưu hoá mô hình làm sẵn . Việc tạo thể hiện và huỷ đối tượng cũng chiếm một chi phí tính toán đáng kể .
  - Thay vì tạo thể hiện và huỷ hết đối tượng này đến đối tượng khác thì một mảng các mô hình đối tượng game làm sẵn sẽ được tạo ra dưới dạng "pool" (bể chứa) để có thể lấy ra đối tượng khi cần rồi trả lại đối tượng khi không sử dụng nữa .

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
- Coroutine:
  - **Corooutine** có thể tạm ngừng việc thực thi của nó trong một frame rồi lấy lại và tiếp tục thực thi trong frame tiếp theo qua bất kỳ độ dài frame nào.
  - Khi bạn gọi một hàm khác từ trong hàm Update() hay FixedUpdate() thì hàm bạn gọi được chạy từ lúc bắt đầu cho tới lúc kết thúc một frame.
  - Ở tốc độ là 60fps thì điều này nghĩa là hàm chạy 60 lần trên giây . Thường thì bạn không cần một hàm phải được gọi trong mọi frame, kể cả khi bạn cần chạy hàm đó định kỳ.
  - Vậy từ nay những hàm mà bạn không cần gọi ở trong tất cả các frame thì bạn hãy sử dụng Coroutine . Thử so sánh một hàm chạy 60 lần trên giây và 10 lần trên giây mà hiệu năng game vẫn vậy thì bạn chọn cách nào , rõ ràng trong trường hợp này dùng Coroutine sẽ tăng hiệu năng lên 6 lần .

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
- Tối ưu hoá với Mecanim:
  - Dưới đây là một số lưu ý để dùng Mecanim hiệu quả hơn .
  - Sử dụng bảng băm thay vì chuỗi để truy vấn Animator.
  - Triển khai AI Layer để điều khiển Animator. Bạn có thể dùng AI Layer để cung cấp các lời gọi callback đơn giản cho hàm OnStateChange(), OnTransitionBegin , ...
  - Sử dụng tag state để dễ dàng ánh xạ máy trạng thái AI với máy trạng thái Mecanim.
  - Sử dụng đường cong phụ trợ để mô phỏng Events .
  - Sử dụng đường cong phụ trợ để đánh dấu animation(ví dụ như kết hợp với việc ánh xạ mục tiêu)

Tối ưu hoá game trong unity

- **Tối ưu hoá khi viết code trong các Script?**
- Tối ưu hoá với Mecanim:
  - Tối ưu hoá thời gian chạy :
  - Luôn tối ưu hoá animation bằng cách thiết lập mục Culling Mode của animator về thành Based on Renderers , và vô hiệu hoá thuộc tính Update when offscreen của trình render lưới da (skinned mesh renderer) . . Theo đó thì cách animation sẽ không phải cập nhật khi nhân vật chưa hiện ra .
  - Một kinh nghiệm nữa là nên dùng ID bảng băm (hash ID) Khi làm việc với animator , việc sử dụng định dạng bảng băm (hash identifier) là một phương pháp tối ưu hoá trong đó tên của các trạng thái và các tham số được gán là kiểu integer . Việc sử dụng kiểu integer thay vì sử dụng chuỗi sẽ giúp giảm chi phí xử lý phụ thêm .

Tối ưu hoá game trong unity

- **Tổng kết:**
- Về giải thuật thì ta code sao cho tối ưu, tránh thừa
- Các object thường xuyên phải sử dụng destroy (như đạn) nên sử dụng dang poolObject
- Giảm thiểu Draw call, Giảm số lượng định và cạnh của các hình được vẽ trong game có thể sử dụng spritePacker

Tối ưu hoá game trong unity

- **Tổng kết:**
- Về **giải thuật** thì ta code sao cho tối ưu, tránh thừa
- Các object thường xuyên phải sử dụng destroy (như đạn) nên sử dụng dang poolObject
- Giảm thiểu Draw call, Giảm số lượng định và cạnh của các hình được vẽ trong game có thể sử dụng spritePacker
- **Đồ họa**: Sử dụng đồ họa atlas/ Sprite Sheet. Có thể hiểu như 1 nhân vật có tay,chân,mắt…. sẽ add vào 1 tấm ảnh chung sử dụng vật liệu chung. 1 số tool hỗ trợ tạo cái này sẽ update kèm theo tài liệu, hoặc cách sử dụng dễ dàng nhất là sử dụng chế độ multiple của Sprite mode. Hoặc tham khảo cách tạo atlas của NGUI.
  - Hạn chế tối đa sử dụng materials (vật liệu) khác nhau giữa các object khác nhau.Sử dụng texture compress là tối ưu nhất và nên sử dụng 16 bit hơn là 32 bit.Nếu có thể thì dùng Animation Clips sẽ tốn ít CPU hơn rất nhiều so với sử dụng Animator.Trên IOS mọi ảnh nên chuyển về **PVRTC**.

Tối ưu hoá game trong unity

- **Tổng kết:**
- **Scripts**
  - Không lạm dụng các hàm *OnGUI()* và *FixedUpdate()* với các vấn đề không liên quan đến GUI vì 2 hàm trên tiêu tốn tài nguyên lớn hơn rất nhiều so với *Update()*. Hàm *FixedUpdate()* có thể gọi tới 50-100 lần trong 1s trên mỗi 1 object nên cẩn thận khi dùng.
  - Tránh viết các hàm *Update()* trống(xóa đi nếu không sử dụng hàm update). Tránh sử dụng các hàm search object.
  - Ví dụ như *GameObject.FindByTag()*, *GameObject.GetComponent()*... Các hàm trên sử dụng dễ dàng nhưng mỗi lần gọi hàm là 1 lần duyệt lại tất cả các object trong scene. Để tối ưu hơn chúng ta nên tạo các biến, đối tượng trong script rồi ném vào ở trong *Inspector*.
  - Không sử dụng *SendMessage()* nếu không cần thiết, hàm này chậm hơn 100 lần so với cách gọi hàm thông thường hoặc thông qua gameobject.
  - Sự dụng InvokeRepeat() thay cho Update() trong một số trường hợp đặc biệt.

Tối ưu hoá game trong unity

- **Tổng kết:**
- **Physics**
  - Sử dụng boxcollider thay vì sử dụng poligon collider, mesh collider..
  - Nên sử dụng Raycast thay vì sử dụng OnTriggerEnter hay Exit… Raycast sẽ nhẹ hơn rất nhiều và mạng lại độ chính xác tuyệt đối. Tất nhiên nhiều trường hợp vẫn phải dùng OnTrigger hoặc OnCollision.
  - Nếu mà đối tượng cha có rigidbody rồi thì đối tượng con chỉ cần colldier thôi chứ không nhất thiết phải thêm rigidbody.
  - Loại bỏ tối đa các hình ảnh có transparent thừa quá nhiều.
  - Chú ý: Xóa hết tất cả các Rigid trong các Object không di chuyển trừ những trường hợp có gắn script lắng

Tối ưu hoá game trong unity

- **Tổng kết:**
- **GameObject**
  - Sử dụng static với các Object tĩnh. Nó được hiểu là các vật thể **không** di chuyển trong game, ví dụ như hòn đá, cái cây, con đường,…. Bất cứ cái gì không dịch chuyển thì ta dùng static. Các bạn có thể tìm thấy nó ở trong Inspector phía trái trên và tích vào đó là được.
  - **Chú ý**: Phải chắc chắn rằng object không dịch chuyển.

Tối ưu hoá game trong unity

- **Tổng kết:**
- **Sound, Audio**
  - Sử dụng Compressed Audio cho các loại âm thanh, nhạc nền.
  - Sử dụng UnCompressed Audio cho các âm thanh kích thước nhỏ, âm hiệu ứng.
  - Sử dụng hợp lý qua lại giữa 3 lựa chọn *Decompress On Load*, *Compressed in memory* và *Stream from disc*.
  - *Decompress On Load* được hiểu là ta sẽ bỏ nén hết các audio khi load. Phương án này được dùng cho các audio nhỏ, nhẹ. Còn nhạc nền hay sound dài thì tránh không được sử dụng nếu không muốn load game 30'

Tối ưu hoá game trong unity

- **Tổng kết:**
- **Sound, Audio**
- *Compressed in memory* được hiểu như là: Máy sẽ giữ tạm bản nén của âm thanh trong bộ nhớ, khi nào chương trình cần thì giải nén ra mà chạy. Phương án này nên được sử dụng với các loại âm thanh lớn.
- *Stream from disc* được hiểu 1 cách rất đơn giản là ta sẽ đọc trực tiếp file âm thanh chứ không cần lưu ở bộ nhớ nữa. Hardware decoding là lựa chọn nhanh nhất và tốt nhất.

Tối ưu hoá game trong unity

- **Tổng kết:**
- **Setting**
- Sử dụng hợp lý 2 Option khi build trên IOS là ***Slow and Safe*** và **Fast and Exceptions Unsuportsed.** Theo kinh nghiệm thì cứ chọn phương án 2 trừ những game nào cần độ chính xác tuyện đối thì nên chọn phương án 1.
- **Orther**
- Những vấn đề cần tối ưu sẽ được hiển thị rất rõ ràng khi chạy game phần Statistics và trong Profiler.
- **Nâng cao**
- Cộng trừ Vector3 sẽ không nhanh và tiêu tốn CPU hơn

Tối ưu hoá game trong unity

- **Tổng kết:**
- Ví dụ về cộng vector3:

```
Vector3 a;
Vector3 b;
Vector c;
c=a+b;
```

- Đoạn mã trên sẽ chậm hơn so với cách dùng sau:

```
Vector3 a;
Vector3 b;
Vector c;
c.x = a.x+b.x;
c.y = a.y+b.y;
c.z = a.z+b.z;
```

Tối ưu hoá game trong unity

- **Tổng kết:**
- Sử dụng Object Caching: hiểu đơn giản là lưu đối tượng hoặc các Component vào 1 biến local sau sử dụng lại. Ví dụ đơn giản: *void Awake(){*

```
MyTransform = transform;
}
void Update(){
//Update vị trí thông qua cache transform
MyTransform.position = MyPosition;
}
```

- Sử dụng 2 hàm rất hay trong Unity đó là *OnBecameVisible ()* và *OnBecameInvisible()* . Hai hàm này rất hay vì tác dụng của nó là gọi khi Object xuất hiện ở bất kỳ camera nào hoặc không xuất hiện

Tăng số lượt download

- **1. Tiêu đề cho game**
  - Tiên đề luôn hướng tới xu hướng tìm kiếm của người dùng, ví dụ một tên tiêu đề tốt “Game Mario 2017 HD”, chứ không phải chỉ là “Game Mario“, 10% tiêu đề quyết định sự xuất hiện trên bộ máy tìm kiếm, lời khuyên của tôi là tìm những ứng dụng Top và học lại cách đặt tên của họ. Đó là cách tốt để đặt tên tiêu đề

Tăng số lượt download

- **2. Từ khóa**
  - Bước đầu tiên là nghiên cứu từ khóa cho game mobile của bạn, ví dụ từ khóa tốt là từ khóa mọi người hay tìm kiếm “Game Mario 2017”, từ 2017 ám chỉ là mốc thời gian cố định, hoặc “Game Mario HD Hay Nhất”, sau khi nghiên cứu từ khóa xong các bạn đưa vào bài viết Description và cố gắng dàn trải từ khóa trong 3 phần, mở đầu thân và cuối.

Tăng số lượt download

- **3. Mô tả game hợp lí**
  - Tạo ra một mô tả hoàn hảo như thể là bạn đang kể một câu chuyện hấp dẫn cho người chơi, hãy tạo những điểm nhấn bằng cách kêu gọi mọi người nhấn nút **G+**, **share Game** và sử dụng ngắt nghỉ một cách hợp lý, bài viết nên có những con số và những mốc thời gian hay là địa điểm…

Tăng số lượt download

- **4. Sử dụng analytics**
  - Analytics là gì các bạn có thể xem thêm tại đây http://www.google.com/analytics/ về cơ bản thì chúng ta có thể hiểu Analytics giúp chúng ta phân tích – kiểm tra – đánh giá được số lượng hành vi người sử dụng của các ứng dụng của bạn, từ đó đưa ra những chiến lược hợp lý cho những bản update tiếp theo
- **5. Icon:** Icon đẹp có thể gây ấn tượng mạnh với người tìm kiếm game, nên có thể năng số lượt download

DEMO

Xây dựng scene GameManager (phần 2)

- Xây dựng game hoàn chỉnh:
- Nhân vật chạy, ăn coin, có điều

Viết code cho file fade.cs

Fade.cs

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Fade : MonoBehaviour
5  {
6      /**
7       *
8       * @author hungnq
9       */
10     GUITexture Black_screen;
11     public float Fade_Time = 2f;
12     public float Fade_Max = 1f;
13     float _time;
14     public bool FadeIn_ing = true;
15     public bool FadeOut_ing;
16
17     void Start ()
18     {
19         Black_screen = GetComponent<GUITexture> ();
20     }
21
22     void Update ()
23     {
24         //Thay đổi màu screen theo trạng thái fade in, fade out
25         if (FadeIn_ing) {
26             _time += Time.deltaTime;
27             Black_screen.color = Color.Lerp (new Color (0, 0, 0, Fade_Max), new Color (0, 0, 0, 0), _time / Fade_Time);
28         }
29
30         if (FadeOut_ing) {
31             _time += Time.deltaTime;
32             Black_screen.color = Color.Lerp (new Color (0, 0, 0, 0), new Color (0, 0, 0, Fade_Max), _time / Fade_Time);
33         }
34
35         if (_time >= Fade_Time) {
36             _time = 0;
37             FadeIn_ing = false;
38             FadeOut_ing = false;
39         }
40     }
```

Viết code cho file fade.cs

```
41  //Định nghĩa hàm FadeIn
42      public void FadeIn ()
43      {
44          FadeIn_ing = true;
45      }
46  //Định nnghĩa hàm FadeOut
47      public void FadeOut ()
48      {
49          FadeOut_ing = true;
50      }
51  }
```

Viết code cho Gui_Layout.cs

Fade.cs | Gui_Layout.cs

```
1  using UnityEngine;
2  using System.Collections;
3
4  [ExecuteInEditMode]
5  public class Gui_Layout : MonoBehaviour
6  {
7      /**
8       *
9       * @author hungnq
10      */
11      //Khai báo các kiểu vị trí của layout tương tác
12     public enum positionType
13     {
14         TopLeft,
15         TopMiddle,
16         TopRight,
17         MiddleLeft,
18         Middle,
19         MiddleRight,
20         BottomLeft,
21         BottomMiddle,
22         BottomRight
23     }
24 //Khai báo các biến cần sử dụng
25     public positionType _positionType = positionType.Middle;
26     public float margin_x;
27     public float margin_y;
28     public int _depth;
29     float screenX;
30     float screenY;
31     GUIText _gui_text;
32     GUITexture _gui_texture;
33     float _guiWidth;
34     float _guiHeight;
35
36     bool TextureIN=false;
37
38     void Awake ()
39     {
40         //Giao diện hiển thị theo trạng thái game
41         #if !(UNITY_EDITOR)
42
43         _gui_text = GetComponent<GUIText> ();
44         _gui_texture = GetComponent<GUITexture> ();
45         screenX = Screen.width;
46         screenY = Screen.height;
```

Viết code cho Gui_Layout.cs

Fade.cs | Gui_Layout.cs

```
47
48
49
50          if (_gui_texture != null) {
51              _guiWidth = _gui_texture.pixelInset.width;
52              _guiHeight = _gui_texture.pixelInset.height;
53          }
54
55          PositionSetting ();
56          #endif
57
58
59      }
60
61      void Update ()
62      {
63          //Update trạng thái sau khi người chơi tương tác
64          #if UNITY_EDITOR
65
66          _gui_text = GetComponent<GUIText> ();
67          _gui_texture = GetComponent<GUITexture> ();
68          screenX = Screen.width;
69          screenY = Screen.height;
70
71
72
73          if (_gui_texture != null) {
74
75              _guiWidth = _gui_texture.pixelInset.width;
76              _guiHeight = _gui_texture.pixelInset.height;
77
78              if(_gui_texture.texture!=null && TextureIN==false){
79                  TextureIN =true;
80                  _guiWidth = _gui_texture.texture.width;
81                  _guiHeight = _gui_texture.texture.height;
82                  TextureIN =false;
83              }
84
85          }
86
87          this.gameObject.transform.position = new Vector3 (0, 0, -0.01f * _depth);
88          PositionSetting ();
89
90          #endif
91      }
```

Viết code cho Gui_Layout.cs

```
92  //Cài đặt vị trí
93      void PositionSetting ()
94      {
95          switch (_positionType) {
96          case positionType.TopLeft:
97
98              if (_gui_text != null)
99                  _gui_text.pixelOffset = new Vector2 (margin_x, screenY - margin_y);
100             if (_gui_texture != null)
101                 _gui_texture.pixelInset = new Rect (margin_x, screenY - _guiHeight - margin_y,_guiWidth, _guiHeight);
102
103             break;
104
105         case positionType.TopMiddle:
106
107             if (_gui_text != null)
108                 _gui_text.pixelOffset = new Vector2 (screenX * 0.5f - margin_x, screenY - margin_y);
109             if (_gui_texture != null)
110                 _gui_texture.pixelInset = new Rect (screenX * 0.5f + margin_x - _guiWidth * 0.5f,
111                     screenY - _guiHeight - margin_y, _guiWidth, _guiHeight);
112
113             break;
114
115         case positionType.TopRight:
116
117             if (_gui_text != null)
118                 _gui_text.pixelOffset = new Vector2 (screenX - margin_x, screenY - margin_y);
119             if (_gui_texture != null)
120                 _gui_texture.pixelInset = new Rect (screenX - margin_x - _guiWidth,
121                     screenY - _guiHeight - margin_y, _guiWidth, _guiHeight);
122
123             break;
124
125         case positionType.MiddleLeft:
126
127             if (_gui_text != null)
128                 _gui_text.pixelOffset = new Vector2 (margin_x, screenY * 0.5f);
129             if (_gui_texture != null)
130                 _gui_texture.pixelInset = new Rect (margin_x, screenY * 0.5f - _guiHeight * 0.5f
131                     + margin_y, _guiWidth, _guiHeight);
132
133             break;
134
```

Viết code cho Gui_Layout.cs

Fade.cs | Gui_Layout.cs

```
135         case positionType.Middle:
136
137             if (_gui_text != null)
138                 _gui_text.pixelOffset = new Vector2 (screenX * 0.5f + margin_x, screenY * 0.5f + margin_y);
139             if (_gui_texture != null)
140                 _gui_texture.pixelInset = new Rect (screenX * 0.5f - _guiWidth * 0.5f + margin_x, screenY * 0.5f
141                     - _guiHeight * 0.5f + margin_y, _guiWidth, _guiHeight);
142
143             break;
144
145         case positionType.MiddleRight:
146
147             if (_gui_text != null)
148                 _gui_text.pixelOffset = new Vector2 (screenX - margin_x, screenY * 0.5f + margin_y);
149             if (_gui_texture != null)
150                 _gui_texture.pixelInset = new Rect (screenX - margin_x - _guiWidth, screenY * 0.5f
151                     - _guiHeight * 0.5f + margin_y, _guiWidth, _guiHeight);
152
153             break;
154
155         case positionType.BottomLeft:
156
157             if (_gui_text != null)
158                 _gui_text.pixelOffset = new Vector2 (margin_x, margin_y);
159
160             if (_gui_texture != null)
161                 _gui_texture.pixelInset = new Rect (margin_x, margin_y, _guiWidth, _guiHeight);
162
163             break;
164
165         case positionType.BottomMiddle:
166
167             if (_gui_text != null)
168                 _gui_text.pixelOffset = new Vector2 (screenX * 0.5f + margin_x, margin_y);
169             if (_gui_texture != null)
170                 _gui_texture.pixelInset = new Rect (screenX * 0.5f + margin_x - _guiWidth * 0.5f,
171                     margin_y, _guiWidth, _guiHeight);
172
173             break;
174
175         case positionType.BottomRight:
176
177             if (_gui_text != null)
178                 _gui_text.pixelOffset = new Vector2 (screenX - margin_x, margin_y);
179             if (_gui_texture != null)
180                 _gui_texture.pixelInset = new Rect (screenX - margin_x - _guiWidth, margin_y, _guiWidth, _guiHeight);
```

Tổng kết

- Tối ưu game
- Phương pháp cải thiện số lượt download

Fpt University

FPT POLYTECHNIC

THANK YOU!

www.poly.edu.vn