**Bìa 1**

**Bìa 2**

# LỜI CAM ĐOAN

# LỜI CẢM ƠN

Để hoàn thành khóa luận tốt nghiệp này em xin chân thành cảm ơn quý thầy cô giáo trong khoa Công Nghệ Thông Tin trường Đại Học Sài Gòn đã tạo điều kiện để em học tập và thực hiện khóa luận.

Đặc biệt, em xin gửi đến thầy Nguyễn Minh Thi, người đã tận tình hướng dẫn, giúp đỡ em hoàn thành khóa luận tốt nghiệp này lời cảm ơn sâu sắc nhất.

Vì kiến thức bản thân còn hạn chế, trong quá trình thực hiện và bảo vệ khóa luận này em không tránh khỏi những sai sót, rất mong nhận được sự thông cảm cũng như những ý kiến đóng góp từ thầy cô.

# MỤC LỤC

[LỜI CAM ĐOAN i](#_Toc498559222)

[LỜI CẢM ƠN ii](#_Toc498559223)

[MỤC LỤC iii](#_Toc498559224)

[DANH MỤC CÁC TỪ VIẾT TẮT iv](#_Toc498559225)

[DANH MỤC CÁC BẢNG iv](#_Toc498559226)

[DANH MỤC CÁC SƠ ĐỒ v](#_Toc498559227)

[**1** **CHƯƠNG 1: TỔNG QUAN VỀ GAME ENGINE VÀ UNITY3D** 1](#_Toc498559228)

[**1.1.** **Game Engine** 1](#_Toc498559229)

[**1.1.1.** **Khái niệm** 1](#_Toc498559230)

[**1.1.2.** **Xu hướng hiện nay** 1](#_Toc498559231)

[**1.2.** **Unity3D** 3](#_Toc498559232)

[**1.2.1.** **Tổng quan** 3](#_Toc498559233)

[**1.2.2.** **Lịch sử hình thành, phát triển và thành tựu đạt được** 3](#_Toc498559234)

[**1.2.3.** **Tại sao là Unity?** 5](#_Toc498559235)

[**1.2.4.** **Các sản phẩm nổi bật** 6](#_Toc498559236)

[**2** **CHƯƠNG 2: Chi tiết về Unity3D** 6](#_Toc498559237)

[**2.1.** **Các thành phần chính** 6](#_Toc498559238)

[**2.1.1.** **Các khái niệm cơ bản** 6](#_Toc498559239)

[**2.1.2.** **Physics** 8](#_Toc498559240)

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **UI** | **User Interface** |
|  |  |

# DANH MỤC CÁC BẢNG

# DANH MỤC CÁC SƠ ĐỒ

2. **CHƯƠNG 1: TỔNG QUAN VỀ GAME ENGINE VÀ UNITY3D**
   1. **Game Engine**
      1. **Khái niệm**

Một game engine là một phần mềm được viết để thiết kế và phát triển video game, hiểu đơn giản nó là loại phần mềm trung gian kết nối tương tác của nhiều ứng dụng trong cùng 1 hệ thống với nhau. Có rất nhiều loại game engine dùng để thiết kế game cho các hệ máy như hệ consoles(Xbox, PlayStation) hay máy tính cá nhân(PC). Chức năng cốt lõi của game engine phần lớn nằm trong công cụ dựng hình(kết xuất đồ họa) cho các hình ảnh 2 chiều(2D) hay 3 chiều(3D), công cụ vật lý (hay công cụ tính toán và phát hiện va chạm), âm thanh, mã nguồn, hình ảnh động (animation), dựng ảnh đồ thị... Quá trình phát triển game tiết kiệm được rất nhiều thời gian và kinh phí vào việc tái sử dụng và tái thích ứng một engine để tạo nhiều game khác nhau.

Chúng ta cũng cần chú ý, game engine không chỉ bao gồm thành phần tạo hình ảnh mà bao gồm rất nhiều thành phần khác như trí tuệ nhân tạo, phân luồng, tạo dòng dữ liệu xử lý, quản lý bộ nhớ, và kết nối mạng… Nếu Engine chỉ có chức năng render hình ảnh thì thường được gọi là Graphic Engine.

* + 1. **Xu hướng hiện nay**

Việc ngày càng có nhiều game engine mới xuất hiện và những game engine cũ được cải tiến thì cuộc cạnh tranh về thị phần nhà phát triển ngày càng gay gắt. Các game engine phải không ngừng tối ưu, cho ra những chức năng mới tiện dụng để hỗ trợ các lập trình viên. Thông thường khi khi chọn lựa một game engine để xây dựng sản phẩm, các nhà phát triển sẽ dựa vào những tiêu chí chủ yếu sau:

* **Giá thành:** Một số game Engine có giá thành cao đi kèm theo đó có rất nhiều chức năng nâng cao và khả năng hỗ trợ tốt. Tuy vậy cũng có những game engine miễn phí nhưng hỗ trợ gần như đầy đủ các tính năng cơ bản. Do vậy chúng ta cần xem xét đến vấn đề này trước tiên. Nếu là người lập trình thông thường thì với mức giá khoảng 1500$(mức giá của Unity3D) cho 1 Engine rõ ràng là một vấn đề lớn.
* **Hiệu năng:** Trước khi chúng ta lựa chọn một Engine, chúng ta cần kiểm tra hiệu năng của Engine này trên nền tảng mà chúng ta hướng đến. Ví dụ: Một engine có hiệu năng cao trên PC nhưng lại cho hiệu năng thấp hoặc không hỗ trợ đầy đủ trên Android. Hoặc 1 Engine chỉ cho 1-2 Frame Per Second(FPS) trong khi 1 Engine khác cho 30 FPS với cùng một cách lập trình.
* **Cộng đồng:** Số lượng người sử dụng rõ ràng nói lên chất lượng của Engine đó. Đồng thời, nhiều người sử dụng và nhiều lập trình viên có nghĩa là khi bạn vướng mắc một vấn đề nào đó, khi đưa câu hỏi lên, sẽ nhanh có câu trả lời hơn, và câu trả lời chất lượng hơn.
* **Tài liệu:** Nếu một Engine không có tài liệu cụ thể, rõ ràng đó là một engine không tốt. Tài liệu cụ thể sẽ giúp bạn rất nhiều khi mới làm quen với Engine, cũng như giải quyết các vấn đề về thiết kế cũng như các vấn đề xảy ra trong khi lập trình.
* **Mã nguồn:** đôi khi các tài liệu không có đủ thông tin, chúng ta cần tìm hiểu mã nguồn để hiểu cách xử lý của Engine. Một số Engine mặc dù Free như lại có mã nguồn đóng. Do đó chúng ta cần có sự lựa chọn. Hoặc đôi khi mã nguồn của Engine không thực sự phù hợp với yêu cầu của chúng ta, khi đó chúng ta cần sửa đổi mã nguồn của Engine đó. Mã nguồn mở cũng giúp cho việc sửa lỗi nhanh hơn do có nhiều người có thể đóng góp hơn.
* **Các tính năng:** Rõ ràng mục đích của chúng ta khi sử dụng Engine là tiết kiệm thời gian phát triển và dễ dàng hơn trong quá trình phát triển. Do đó một Engine có nhiều tính năng hơn sẽ giúp chúng ta nhiều hơn, phù hợp hơn trong nhiều ứng dụng.
* **Store:** việc phát triển từ những gì có sẵn bao giờ cũng ít tốn thời gian và công sức hơn làm tất cả từ đầu. Vậy nên việc có một cộng đồng trao đổi và mua bán asset lớn mạnh sẽ giúp cải thiện đáng kể thời gian phát triển sản phẩm.
  1. **Unity3D**
     1. **Tổng quan**

Unity 3D Game là một engine game đa nền tảng được phát triển bởi Unity Technologies bằng ngôn ngữ C/C++, hỗ trợ mã viết bằng C#, JavaScript hoặc Boo. Đây là công cụ phát triển game được cộng đồng làm game ưa thích sử dụng. Hiểu đơn giản Unity 3D Game là một công cụ trung gian tương tác giữa thiết kế và lập trình để phát triển game, gắn kết với nhiều ứng dụng với nhau trong cùng 1 hệ thống. Unity 3D hỗ trợ tốt cho game phát triển trên nhiều nền tảng khác nhau, có thể hoạt động trên nhiều nhóm thiết bị, từ trình duyệt web trên PC cho tới điện thoại hay máy tính bảng.

* + 1. **Lịch sử hình thành, phát triển và thành tựu đạt được**
* **Lịch sử hình thành và phát triển**
* Vào đầu những năm 2000, ba lập trình viên trẻ với nguồn kinh tế eo hẹp David Helgason (CEO), Nicholas Francis (CCO), and Joachim Ante (CTO) đã tập trung tại một tầng hầm ở Copenhagen và bắt đầu lập trình ra những thứ cơ bản nhất mà sau này trở thành một trong những phần mềm được ứng dụng rộng rãi nhất trong ngành công nghiệp video game.
* Một phiên bản đơn giản của Unity đã được phát hành vào năm 2005. Đội đã có thể hỗ trợ cho máy tính Windows và các trình duyệt Web. Đến năm 2008, engine đã trở nên phức tạp hơn và doanh số bán phần mềm đã bắt đầu chi trả được cho các chi phí hoạt động và cho phép đội phát triển Unity mở rộng đến hàng chục nhân viên.
* Một bước ngoặt đã đến vào giữa năm 2008 khi Apple công bố App Store. "Chúng tôi đã gấp rút xử lý để hỗ trợ iPhone và trở thành game engine đầu tiên làm được điều đó vào cuối năm 2008" - Helgason nói - "Nó xảy ra thực sự nhanh chóng. Đột nhiên, có rất nhiều người muốn có Unity."
* Một bước tiến lớn nữa đã đến vào năm 2008 là khi Cartoon Network sử dụng Unity3D để tạo ra FusionFall, một MMORPG cho trẻ em đã có được 8 triệu người chơi.
* Electronic Arts sử dụng Unity3D trong năm 2009 để làm Tiger Woods PGA Tour Online, và thậm chí cả Microsoft và Ubisoft đã trở thành khách hàng của Unity3D.
* Trong năm 2011, Unity đã mua một công ty hoạt hình có tên Mecanim, thúc đẩy công nghệ cơ bản của game engine. Unity đã mua lại Applifier, một nhà cung cấp dịch vụ di động dựa vào Helsinki vào tháng 3 năm 2014. Hai vụ mua lại tiếp theo vào cuối năm 2014: Playnomics, một nền tảng phân tích dữ liệu cho các nhà phát triển (nay là Unity Analytics) và Tsugi, người đã từng làm việc cho một dịch vụ tích hợp liên tục, bây giờ được gọi là Unity Cloud Build.
* **Những thành tựu đạt được**
* Vào tháng 4 năm 2012, Unity cho biết có 1 triệu nhà phát triển đã đăng ký, 300.000 trong số đó đã sử dụng Unity hàng tháng. Vào tháng 4 năm 2015, số lượng các nhà phát triển đã đăng ký báo cáo đạt 4,5 triệu, với 1 triệu người dùng hoạt động hàng tháng. 47% các nhà phát triển trò chơi điện thoại di động sử dụng Unity.
* Vào ngày 10 tháng 11 năm 2010, Unity Asset Store được đưa ra như một thị trường trực tuyến cho người dùng Unity bán các asset của dự án (ảnh nghệ thuật, các đoạn mã, âm thanh, v.v.) cho nhau. Đến tháng 4 năm 2014, đã có 600.000 người dùng đăng ký tải xuống khoảng 500.000 asset mỗi tháng.
* Tại hội nghị VR/AR với mục đích "mở rộng cơ sở tri thức của bất kỳ ai phát triển thực tế ảo và thực tế tăng cường" được Unity tổ chức vào tháng 2/2016, công ty đã thông báo hợp tác với Steam VR và Google Cardboard và thành lập bộ phận VR/AR của công ty, Unity Labs.
* Từ đó đến nay Unity vẫn đang dẫn đầu về công nghệ VR và chiếm một thị phần rất lớn trên hai thiết bị phổ thông nhất hiện nay: 90% đối với Samsung Gear VR và 53% đối với Oculus Rift. Còn về AR thì unity vẫn là lời khuyên số một để phát triển các ứng dụng cho Microsoft Hololen, tích hợp Vuforia(một trong những tool phát triển VR nổi tiếng nhất hiện nay), hỗ trợ các nền tảng mới ARKit(Apple) và ARCore(Google) ngay khi vừa được ra mắt.
* Đến nay Unity đã hỗ trợ trên 27 nền tảng bao gồm: Android, Android TV, Facebook Gameroom, Fire OS, Gear VR, Google Cardboard, Google Daydream, HTC Vive, iOS, Linux, macOS, Microsoft HoloLens, Nintendo 3DS family, Nintendo Switch, Oculus Rift, PlayStation 4, PlayStation Vita, PlayStation VR, Samsung Smart TV, Tizen, tvOS, WebGL, Wii U, Windows, Windows Phone, Windows Store, và Xbox One
  + 1. **Tại sao là Unity?**
* **Chi phí thấp:** Theo chính sách của Unity Technology, những cá nhân và doanh nghiệp có doanh thu dưới 100.000 USD/năm được dùng miễn phí engine Unity 3D, và chỉ thu phí 1.500 USD/năm cho bản Profestional, một con số rất khiêm tốn so với những gì engine này mang lại.
* **Cộng đồng rất lớn mạnh:** Là engine phổ biến nhất trên thế giới nên Unity có cộng đồng sử dụng rất lớn mạnh. Mọi thắc mắc của bạn về Unity đều sẽ được thảo luận và trả lời trên website cộng đồng http://answers.unity3d.com. Các tân binh sẽ dễ dàng kế thừa, phát triển các scripts và kho đồ họa có sẵn để giúp cho sản phẩm phong phú hơn.
* **Dễ dàng tiếp cận:** theo đánh giá của các lập trình viên thì editor của unity là một điểm tích cực, giao diện cực kì trực quan và dễ sử dụng. Bên cạnh đó là cách kết hợp các component và script giúp lập trình viên dễ dàng hiểu được cách vận hành của hệ thống.
* **Dễ dàng học hỏi:** tài liệu để học trên trang web của unity được viết một cách rất bài bản từ mô tả cho đến các thuộc tính, đi kèm với đó là các ví dụ minh họa về sử dụng scripts hay các hướng dẫn thực hiện theo từng bước có cả ảnh chụp màn hình để minh họa. Ngoài ra còn có các chuỗi bài học với nhiều cấp độ từ những người mới bắt đầu cho đến những nhà phát triển đã có kinh nghiệm.
* **Cập nhật thường xuyên:** Unity 2017.2 là phiên bản mới nhất của được Unity giới thiệu vào tháng 10 năm 2017. Những tính năng ấn tượng được giới thiệu trong phiên bản mới này tạo thêm niềm tin tưởng vững chắc và sức mạnh khó vượt qua được từ Unity.
  + 1. **Các sản phẩm nổi bật**

1. **CHƯƠNG 2: Chi tiết về Unity3D**
   1. **Các thành phần chính**
      1. **Các khái niệm cơ bản**

* **Assets(tài sản)**: Asset là nơi chứa tất cả các file trong Game như: meshes, textures, videos, animations, sounds, music...... Các folder, file có thể tự sắp xếp theo cách của riêng của người dùng làm sao để dễ dàng quản lý nhất, nên thông thường được đặt tên theo các chức năng hoặc ý nghĩa của chúng. Khi muốn đặt một mô hình 3D hay đơn giản chỉ là một tập tin cấu hình xml bên trong Game của bạn, chúng ta sẽ cần phải khai báo(import) một cách đơn giản là kéo thả tập tin vào bên trong dự án để có thể sử dụng như là một asset của dự án. Sau khi được khai báo thì Unity sẽ tạo ra một file .meta có cùng tên với asset để quản lí. Các phương pháp trước đây chỉ cho phép nhập một file tại một thời gian, trong khi sau này cho phép nhiều asset được gom lại thành một gói(package) và khai báo cùng một lúc. Các asset do chính những nhà phát triển game tạo ra và có thể được download miễn phí hoặc trả phí trên Unity Asset Store. Đây là một trong những tính năng rất hay của Unity. Các asset này sẽ giúp giảm thiểu rất nhiều thời gian cho việc thiết kế và lập trình game.

|  |
| --- |
| Học lập trình game cơ bản |
| Hình 1.1: Các loại file được Unity3D hỗ trợ |

* **Scenes**: Trong Unity, một cảnh chơi (hoặc một phân đoạn) là những màn chơi riêng biệt, một khu vực trong game hoặc thành phần có trong nội dung của trò chơi (các menu). Các thành phần này được gọi là Scene. Bằng cách tạo ra nhiều Scenes, chúng ta có thể phân phối thời gian và tối ưu tài nguyên, kiểm tra các phân đoạn trong game một cách độc lập.
* **Game Objects:** Đây là một thuật ngữ thông dụng, đặc biệt trong mảng lập trình game. Mọi cảnh chơi, mọi kịch bản đều được xây dựng lên từ các gameobject và tương tác giữa chúng. Tất cả các Game Object đều chứa ít nhất một thành phần cơ bản là transform, lưu trữ thông tin về vị trí, góc xoay và tỉ lệ của gameobject. Thành phần transform này có thể được tuỳ biến và chỉnh sửa trong quá trình lập trình.
* **Components**: là các thành phần được đính vào trong gameobject, bổ sung tính năng cho các gameobject. Mỗi component có chức năng riêng biệt. Đa phần các component phụ thuộc vào một transform, vì nó lưu trữ các thông số cơ bản của gameobject. Bản chất của gameobject thật sự chỉ là một đơn vị để chứa các component, các đặc tính và khả năng của gameobject nằm hoàn toàn trong các component. Do đó chúng ta có thể xây dựng nên bất kỳ Game Object nào trong game mà chúng ta có thể tưởng tượng được bằng cách đính vào đó những component cần thiết.
* **Script**: script được Unity xem như một Component. Đây là thành phần thiết yếu trong quá trình phát triển game. Bất kỳ một game nào, dù đơn giản nhất đều cần đến script để tương tác với các thao tác của người chơi, hoặc quản lý các sự kiện để thay đổi chiều hướng của game tương ứng với kịch bản game. Unity cung cấp cho lập trình viên khả năng viết script bằng ba ngôn ngữ: JavaScript, C#, Boo. Thông thường tất cả các hoạt động của gameobject hay sâu hơn là các component đều được điều khiển bởi script, điều này tạo nên một sự linh động rất lớn cho quá trình lâp trình.
* **Prefabs**: Prefabs thực chất là một gameobject được lưu trữ lại để tái sử dụng. Các gameobject được nhân bản từ một prefab sẽ giống nhau hoàn toàn, ngoại trừ thành phần Transform và các tham chiếu ra bên ngoài prefab để phân biệt và quản lý được tốt hơn. Nhờ prefab ta có thể tận dụng tối đa những thành phần có thể tái sử dụng để giảm khối lượng công việc khi phát triển game.
  + 1. **Physics**

Để có được hành vi vật lý thuyết phục trong game, đối tượng trong trò chơi phải tăng tốc chính xác và bị ảnh hưởng bởi va đập, lực hấp dẫn và các lực khác. Hệ vật lý tích hợp của Unity sẽ cung cấp một cách đầy đủ các thành phần xử lý mô phỏng vật lý. Chỉ cần một số thiết lập tham số có thể tạo ra các đối tượng hoạt động theo cách thực tế (nghĩa là chúng sẽ bị di chuyển bởi va chạm và rơi…). Bằng cách kiểm soát vật lý từ các script, bạn có thể cho một đối tượng động lực của một chiếc xe, một cái máy, hoặc thậm chí một miếng vải.

**Lưu ý:** có hai hệ vật lý riêng biệt trong Unity: một cho vật lý 3D, và một cho vật lý 2D. Các khái niệm chính giống hệt nhau giữa hai hệ vật lí nhưng chúng được thực hiện bằng các component khác nhau.

* + - 1. **Rigidbody**

Một Rigidbody là thành phần chính xử lí hành vi vật lý của một đối tượng. Với một Rigidbody đính kèm, đối tượng sẽ ngay lập tức đáp ứng những tính chất của trọng lực. Nếu một hoặc nhiều thành phần collider cũng được thêm vào, đối tượng sẽ phát hiện được va chạm và bị tác động từ lực của các va chạm.

Vì một thành phần của rigidbody chiếm quyền điều khiển transform đối tượng mà nó được gắn liền nên không thể di chuyển nó bằng cách thay đổi vị trí và độ xoay như thường lệ. Thay vào đó, nên áp dụng các lực vật lí để di chuyển đối tượng đó hoặc có thể tác động đến rigidbody để dễ kiểm soát hơn.

Trong một số trường hợp, nếu muốn một đối tượng có một rigidbody mà không muốn chuyển động của nó bị kiểm soát bởi các tác động vật lý. Ví dụ: muốn kiểm soát hành động vật lí của nhân vật trực tiếp từ script nhưng vẫn cho phép nó phát hiện được các va chạm của hệ vật lí. Kiểu chuyển động phi vật lý này được gọi là chuyển động kinematic. Thành phần rigidbody có một thuộc tính được gọi là IsKinematic loại bỏ nó khỏi sự kiểm soát của các tác động vật lý và cho phép nó được điều khiển chuyển động từ script. Có thể thay đổi giá trị kinematic từ script để cho phép vật lý được bật và tắt cho một đối tượng, nhưng điều này đi kèm với việc tiêu tốn hiệu năng xử lí nên cần cân nhắc trước khi được sử dụng.

* + - 1. **Colliders**

Như đã đề cập ở trên, collider là thành phần đề phát hiện va chạm trong hệ vật lí của Unity. Vì vậy collider xác định hình dạng theo ý muốn của một vật thể với mục đích phát hiện va chạm vật lý. Một collider không nhìn thấy được và không nhất thiết phải có hình dạng tương tự như của vật thể trên thực tế, trong đa số trường hợp, chỉ cần một collider có hình dáng đơn giản và gần giống như vật thể là đảm bảo tính logic và hiệu suất của ứng dụng.

Một số collider thông dụng trong môi trường 3D như collider hình hộp 3D, hình cầu, hình trụ bo tròn. Trong 2D, có các loại như hình hộp 2D, hình tròn. Ngoài ra, trong 2D Unity cũng hỗ trợ tùy biến colider2D theo các hình dáng phức tạp. Đối với các mô hình 3D xuất ra từ các phần mềm đồ họa thì đã kèm theo meshcollider chính xác theo hình dáng của mô hình.

Colliders có thể được thêm vào một đối tượng mà không cần thành phần Rigidbody để tạo ra các thành phần bất động của game như tường, sàn... Nhìn chung, không nên định vị lại các collider tĩnh bằng cách thay đổi vị trí vì nó sẽ tác động mạnh mẽ đến hiệu năng của hệ vật lý vì nó không có rigidbody nên hệ vật lí không thể tính toán va chạm và các lực khác khi tác động.

* + 1. **UI**

Mặc dù mang tên chính thức là Unity3D nhưng không vì thế mà unity bớt đi sự tập trung và trau chuốt cho mảng 2D của game engine này. Bằng chứng là các thành phần để phát triển game 2D như physic2D và đa nền tảng không thua kém gì mảng 3D. Phần UI cũng không phải là ngoại lệ, UI của unity cung cấp đầy đủ các thành phần cần thiết để xây dựng giao diện người dùng. Bên cạnh đó là khả năng tùy biến vô giới hạn các thành phần, ví dụ như có thể biến một hình ảnh thành một nút bấm bằng cách thêm vào cho nó một event trigger component để tương tác với các sự kiện.

* + - 1. RectTransform

Rect Transform là một component được dùng để thay thế cho thành phần transform trên tất cả các thành phần UI. RectTransform thể hiện một hình chữ nhật được xác định bởi chiều rộng và chiều cao liên quan đến một điểm tâm của nó (gọi là pivot).

Pivot là điểm trụ của object, thông thường mặc định sẽ là ở tâm của hình chữ nhật, nếu xoay hoặc phóng to thu nhỏ RectTransform thì nó sẽ dựa trên điểm tâm này.

Thêm một thuật ngữ mà chúng ta cũng cần phải lưu ý là “neo”(anchor), một RectTransform có thể được neo tới đối tượng cha của nó nếu cha của nó cũng có một thành phần RectTransform. Anchor cho phép chúng ta di chuyển và điều chỉnh kích thước của thành phần mà nó đính kèm theo vị trí và kích thước của UI cha.

* + - 1. Canvas và panel

Canvas và panel là hai thành phần chính để mô hình tổng thể layout người dùng.

* **Canvas:** có thể hiểu là một vùng riêng cho phép chứa đựng thành phần giao diện người dùng (UI). Như vậy, khi tạo ra một UI mới, một Canvas cũng tự động được tạo ra, nếu đã có một đối tượng Canvas rồi thì UI mới được tạo ra là đối tượng con của Canvas. Một trong những mục đích quan trọng nhất của Canvas là để xử lý vấn đề tương thích đa màn hình, với thời đại smart phone hiện nay, có rất nhiều màn hình với tỷ lệ khác nhau, Canvas sẽ tự động scale canh chỉnh lại để duy trì UI của chúng ta được hiển thị tốt, không bị mất đi khi build lên những kích thước màn hình khác nhau.
* **Panel:** cơ bản panel là một thành phần để nhóm các thành phần UI liên quan lại với nhau sau đó mới canh chỉnh pannel này với canvas tổng để không bị vỡ khi chuyển đổi kích thướt màn hình.
  + - 1. Các thành phần cơ bản khác
    1. **Audio và video**
  1. **Giao diện người dùng**
  2. **Làm quen với các thao tác cơ bản**