Buổi 0 – Thứ 2

1. **Unity Engine:**
   1. **Game Engine là gì?**

* Game engine là **một phần mềm** được sử dụng **để phát triển các trò chơi điện tử**.
* Nó **cung cấp** một **cơ sở hạ tầng** để **xây dựng các trò chơi** và **giúp** cho các nhà phát triển dễ dàng **quản lý** **các phần** khác nhau **của một trò chơi**, chẳng hạn như đồ họa, âm thanh, vật lý, AI, hệ thống địa hình, và quản lý tài nguyên.
* Game engine thường **cung cấp** một **công cụ quản lý** các **đối tượng** trong trò chơi, **cách** để **tạo ra các sự kiện**, **lập trình AI** và các **quy tắc chơi game**.
* Game engine cũng thường **cung cấp** các **khả năng** để **xuất sản phẩm** của bạn cho **nhiều nền tảng khác nhau**, chẳng hạn như máy tính, console, và thiết bị di động.
  1. **Real-time Engine là gì?**
* Real-time engine là **một loại phần mềm engine** được sử dụng **để tạo ra các ứng dụng đa phương tiện**, như các trò chơi, các ứng dụng thực tế ảo (virtual reality) hay thực tế tăng cường (augmented reality), các bộ phim và hình ảnh động, các ứng dụng giáo dục, thiết kế, xây dựng, v.v.
* Nó là một **nền tảng phần mềm chuyên biệt**, có khả năng **xử lý các tác vụ đồ họa và tính toán trong thời gian thực (real-time)**, giúp **tạo ra các trải nghiệm tương tác và động** đáng kinh ngạc cho người dùng. Các tính năng của real-time engine bao gồm **khả năng tạo và sửa đổi các đối tượng 3D**, **vật liệu, ánh sáng và hiệu ứng đặc biệt, cũng như điều khiển các tác vụ đồ họa, vật lý và âm thanh trong thời gian thực**.
* Một số real-time engine phổ biến hiện nay bao gồm Unreal Engine, Unity, CryEngine, Lumberyard, v.v. Real-time engine được sử dụng rộng rãi trong ngành công nghiệp trò chơi điện tử và đang ngày càng được ứng dụng trong các lĩnh vực khác như giáo dục, y tế, kiến ​​trúc, v.v.

1. **Version control – git - smartgit**

**Lý Thuyết:**

* 1. **Version control là gì?**
* Version control là **quá trình quản lý và theo dõi** các **thay đổi trong mã nguồn** hoặc tài liệu **trong quá trình phát triển phần mềm** hoặc các dự án khác.
* Nó cho phép **lưu trữ các phiên bản khác nhau của mã nguồn** và tài liệu và **giúp đảm bảo** rằng **các thành viên** trong nhóm **làm việc** có thể **đồng bộ hóa** **các thay đổi** và **tránh ghi đè** lên các thay đổi của nhau.
* Version control cũng **cho phép** các nhà phát triển **quay trở lại phiên bản cũ hơn** của mã nguồn hoặc tài liệu nếu cần thiết, để giải quyết các vấn đề hoặc lỗi phát sinh trong quá trình phát triển. Một số hệ thống version control phổ biến hiện nay bao gồm Git, SVN và Mercurial.
  1. **Git là gì?**
* Git là một hệ thống quản lý phiên bản phân tán (distributed version control system - DVCS), được sử dụng phổ biến trong lĩnh vực phát triển phần mềm. Git cho phép các nhà phát triển làm việc cùng nhau trên cùng một mã nguồn (source code), quản lý và theo dõi các thay đổi trong mã nguồn theo thời gian, từ đó phục vụ cho việc quản lý, phân phối và tích hợp mã nguồn dễ dàng hơn.
* Mỗi lần thay đổi được thực hiện trong mã nguồn, Git sẽ lưu trữ lại thông tin về các thay đổi đó và gán một phiên bản mới cho mã nguồn. Nhờ đó, Git cho phép nhà phát triển theo dõi lịch sử thay đổi, phục hồi các phiên bản trước đó của mã nguồn nếu cần thiết, tạo các nhánh (branch) để phát triển song song các tính năng khác nhau mà không ảnh hưởng đến nhau, và hợp nhất (merge) các nhánh khi cần thiết.
* Git là một công cụ mạnh mẽ và phổ biến, được sử dụng rộng rãi trong cộng đồng phát triển phần mềm, và cung cấp cho người dùng nhiều tính năng linh hoạt và tiện ích để quản lý và theo dõi mã nguồn của mình.
  1. **Smartgit là gì?**
* SmartGit là một ứng dụng desktop đa nền tảng (Windows, macOS, Linux) được sử dụng để quản lý và kiểm soát phiên bản cho các dự án phát triển phần mềm sử dụng Git. Nó cung cấp các tính năng để thực hiện các hoạt động Git phổ biến, bao gồm commit, pull, push, merge, rebase và nhiều tính năng khác. SmartGit cung cấp giao diện đồ họa và dòng lệnh, giúp người dùng tương tác với kho lưu trữ của Git một cách dễ dàng và hiệu quả.
  1. **File readme.md dùng để làm gì?**
* File README.md là một tệp tin dùng để cung cấp thông tin mô tả về dự án, sản phẩm, thư viện, hoặc ứng dụng phần mềm của bạn. Nó thường được đặt ở thư mục gốc của dự án và được hiển thị trực tiếp trên trang GitHub khi truy cập vào thư mục đó.
* README.md thường chứa những thông tin cơ bản về dự án như:
  + Tên dự án
  + Mô tả về dự án
  + Cách cài đặt và sử dụng dự án
  + Hướng dẫn đóng góp và phát triển
  + Liên hệ với tác giả hoặc nhóm phát triển
* Nó là một phần quan trọng của một dự án phần mềm, giúp người dùng và độc giả hiểu rõ hơn về sản phẩm của bạn và cách sử dụng nó.
  1. **File .gitignore dùng để làm gì?**
* File .gitignore là một file cấu hình của Git, dùng để chỉ định các tệp và thư mục mà Git không nên theo dõi (ignore) trong quá trình quản lý phiên bản.
* Khi Git bắt đầu quản lý phiên bản của một thư mục, nó sẽ theo dõi tất cả các tệp tin và thư mục trong thư mục đó. Tuy nhiên, trong một số trường hợp, chúng ta không muốn các tệp tin hoặc thư mục này bị Git theo dõi, ví dụ như các tệp tin tạm thời, các tệp tin được tạo ra bởi các công cụ cụ thể hoặc các thư mục được tạo ra bởi quá trình biên dịch. Khi đó, chúng ta có thể sử dụng file .gitignore để chỉ định cho Git biết những tệp và thư mục nào nên bị bỏ qua trong quá trình quản lý phiên bản.
* File .gitignore thường được đặt ở thư mục gốc của dự án, và có thể chứa các pattern (mẫu) để xác định những tệp tin và thư mục cần bỏ qua. Các pattern này có thể là đường dẫn đầy đủ của các tệp tin hoặc thư mục, hoặc có thể sử dụng các ký tự đại diện để chỉ định một nhóm các tệp tin hoặc thư mục cần bỏ qua. Ví dụ:
* Sử dụng file .gitignore sẽ giúp cho quá trình quản lý phiên bản được sạch sẽ và hiệu quả hơn, giảm thiểu những tệp tin không cần thiết bị đưa vào kho lưu trữ, tăng tốc quá trình tải lên, xuống và đồng bộ hóa dữ liệu.
  1. **Git branch là gì?**
* Trong Git, một branch (nhánh) là một phiên bản của repository, cho phép bạn làm việc song song trên nhiều tính năng, thay đổi và sửa lỗi mà không ảnh hưởng đến phiên bản chính (master branch). Branch giúp người phát triển có thể phát triển tính năng mới hoặc sửa lỗi mà không làm ảnh hưởng đến các tính năng khác.
* Khi bạn tạo một branch mới, Git sẽ tạo ra một bản sao của toàn bộ repository và lưu trữ tại branch mới đó. Sau đó, bạn có thể thêm, sửa hoặc xóa các tệp và commit trên branch đó mà không làm thay đổi các tệp trên các branch khác.
* Sau khi hoàn thành tính năng hoặc sửa lỗi trên branch mới, bạn có thể sử dụng Git merge để hợp nhất các thay đổi từ branch mới vào master branch hoặc branch khác.
  1. **Git clone là gì?**
* git clone là một lệnh trong Git để sao chép một repository (kho chứa code) đang tồn tại trên Git và tạo ra một bản sao trên máy tính cá nhân. Lệnh này cũng tự động thiết lập một kết nối giữa repository trên Git và repository trên máy tính cá nhân để có thể đồng bộ các thay đổi giữa hai repository này.
* Khi sử dụng lệnh git clone, Git sẽ tạo ra một bản sao của toàn bộ lịch sử thay đổi của repository trên Git và đặt nó trong thư mục làm việc hiện tại trên máy tính cá nhân. Sau đó, bạn có thể làm việc với repository đó, thực hiện các thao tác như commit, push, pull, merge, ... giống như với một repository bình thường trên máy tính của bạn.