1. **Trình độ hiện tại của bạn**:

* Bạn đã biết lập trình chưa? Nếu có, bạn quen thuộc với ngôn ngữ nào (C#, Java, Python, v.v.)?
  + Tôi là sinh viên ngành công nghệ thông tin và có kiến thức nền tảng về java ( nhưng chưa vững)
* Bạn đã từng làm quen với Unity hoặc bất kỳ công cụ làm game nào khác chưa?
  + Tôi chưa từng làm quen với Unity hoặc bất kỳ công cụ làm game nào khác

1. **Mục tiêu cụ thể**:

* Bạn muốn phát triển loại game nào (2D, 3D, game mobile, game PC, v.v.)?
  + Tôi vẫn chưa xác định được bản thân muốn phát triển loại game nào. Bạn có thể tư vấn giúp tôi được không? Xu hướng game nào sẽ thu hút nhiều người chơi, được nhiều công ty trong và ngoài nước Việt Nam tuyển dụng?
* Bạn học Unity để làm game cá nhân hay làm việc chuyên nghiệp?
  + Tôi muốn làm project cá nhân trước để xây dựng profile để ứng tuyển việc làm

1. **Thời gian và cam kết**:

* Bạn có thể dành bao nhiêu giờ mỗi ngày hoặc mỗi tuần để học?
  + Mỗi ngày tôi có thể dành 2-3 giờ để học ( có thể lâu hơn nếu cần thiết ), phụ thuộc 1 phần vào lộ trình bạn lập cho tôi, nếu lộ trình cần nhiều thời gian thì bắt buộc tôi phải dành nhiều thời gian để học. Quan trọng là cần có đủ kiến thức, để làm project và chuẩn bị cho tuyển dụng, tôi còn thời hạn khoảng 12 tháng nữa là sẽ kết thúc chương trình học trường đại học, và tôi muốn lúc đó tôi có project đủ xịn để xin ứng tuyển trong các công ty.
* Bạn muốn hoàn thành mục tiêu trong bao lâu?
  + 3- 6 tháng

1. **Nguồn lực có sẵn**:

* Bạn có máy tính đủ mạnh để chạy Unity không?
  + Đủ nha
* Bạn có khả năng tham gia các khóa học trả phí hay chỉ muốn học qua tài liệu miễn phí?
  + Tôi sẵn sàng trả phí học nếu khóa học đó đủ chất lượng, tài chính tôi chỉ đủ 10 triệu.

Tôi muốn học kiến thức mà kiến thức đó sẽ giúp tôi có vị trí mà được nhiều công ty trong và ngoài nước tuyển dụng

**Mục Lục**

[1. Xu hướng game và vị trí tuyển dụng 4](#_Toc189810045)

[1.1 Game 2D: 4](#_Toc189810046)

[1.2 Game 3D: 4](#_Toc189810047)

[2. Xu hướng: 4](#_Toc189810048)

[3. Lộ trình học Unity trong 6 tháng 5](#_Toc189810049)

[3.1 Giai đoạn 1: Nền tảng cơ bản (1-1.5 tháng) 5](#_Toc189810050)

[3.1.1 Học C# cơ bản và ứng dụng trong Unity: 5](#_Toc189810051)

[3.1.2 Làm quen Unity: 5](#_Toc189810052)

[3.1.3 Thực hành mini-project: 5](#_Toc189810053)

[3.2 Giai đoạn 2: Phát triển kỹ năng nâng cao (1.5-2 tháng) 5](#_Toc189810054)

[3.2.1 Học kỹ năng nâng cao: 5](#_Toc189810055)

[3.2.2 Làm game 2D nâng cao: 5](#_Toc189810056)

[3.3 Giai đoạn 3: Học làm game 3D (1.5-2 tháng) 5](#_Toc189810057)

[3.4 Giai đoạn 4: Xây dựng project hoàn chỉnh (1-2 tháng) 6](#_Toc189810058)

[4. Chi tiết các giai đoạn học 8](#_Toc189810059)

[4.1 Giai đoạn 1: Nền tảng cơ bản (1-1.5 tháng) 8](#_Toc189810060)

[4.1.1 Tuần 1-2: Học C# cơ bản và ứng dụng trong Unity 8](#_Toc189810061)

[4.1.2 Tuần 3: Làm quen Unity Editor 9](#_Toc189810062)

[4.1.3 Tuần 4-5: Mini-project game 2D 10](#_Toc189810063)

[4.2 Giai đoạn 2: Phát triển kỹ năng nâng cao (1.5-2 tháng) 12](#_Toc189810064)

[4.2.1 Tuần 1-2: Hiểu sâu về Unity và C# nâng cao 12](#_Toc189810065)

[4.2.2 Tuần 3-4: Làm game 2D nâng cao 14](#_Toc189810066)

[4.3 Giai đoạn 3: Học làm game 3D (1.5-2 tháng) 16](#_Toc189810067)

[4.3.1 Tuần 1: Làm quen với Unity 3D 16](#_Toc189810068)

[4.3.2 Tuần 2: Làm việc với camera và ánh sáng 16](#_Toc189810069)

[4.3.3 Tuần 3-4: Tạo game 3D cơ bản (Dự án nhỏ) 17](#_Toc189810070)

[4.3.4 Tuần 5-6: Làm game 3D nâng cao (Dự án thứ hai) 18](#_Toc189810071)

[4.4 Giai đoạn 4: Xây dựng project hoàn chỉnh (1-2 tháng) 20](#_Toc189810072)

[4.4.1 Tuần 1: Lên ý tưởng và lập kế hoạch chi tiết 20](#_Toc189810073)

[4.4.2 Tuần 2-3: Phát triển dự án 21](#_Toc189810074)

[4.4.3 Tuần 4-5: Hoàn thiện và tối ưu hóa 22](#_Toc189810075)

[4.4.4 Tuần 6: Chuẩn bị phát hành và tạo portfolio 22](#_Toc189810076)

[4.5 Giai đoạn 5: Dự án hoàn chỉnh và chuẩn bị cho tuyển dụng (1.5-2.5 tháng) 24](#_Toc189810077)

[4.5.1 Tuần 1-2: Lập kế hoạch và triển khai dự án hoàn chỉnh 24](#_Toc189810078)

[4.5.2 Tuần 3-4: Phát triển và hoàn thiện dự án 25](#_Toc189810079)

[4.5.3 Tuần 5-6: Hoàn thiện dự án và tạo portfolio 25](#_Toc189810080)

[5. Nội dung bài học 27](#_Toc189810081)

[5.1 Giai đoạn 1: Nền tảng cơ bản 27](#_Toc189810082)

[5.1.1 Tuần 1 -2: Học C# cơ bản và ứng dụng trong Unity 27](#_Toc189810083)

|  |
| --- |
| **Mục lục Nội dung bài học**  [5. Nội dung bài học 31](#_Toc189810668)  [5.1 Giai đoạn 1: Nền tảng cơ bản 31](#_Toc189810669)  [5.1.1 Tuần 1 -2: Học C# cơ bản và ứng dụng trong Unity 31](#_Toc189810670)  [5.1.1.1 Cấu trúc cơ bản của C# 31](#_Toc189810671)  [5.1.1.2 Hàm và lớp ( OPP cơ bản) 34](#_Toc189810672)  [5.1.1.3 Khái niệm OOP ( Lập trình hướng đối tượng ) 34](#_Toc189810673)  [5.1.1.4 Delegate và Event 37](#_Toc189810674)  [5.1.1.4.1 Delegate là gì? 38](#_Toc189810675)  [5.1.1.4.1.1 Cách khai báo và sử dụng delegate 38](#_Toc189810676)  [5.1.1.4.1.2 Xóa một phương thức khỏi delegate 40](#_Toc189810677)  [5.1.1.4.2 Event trong C# 41](#_Toc189810678)  [5.1.1.4.2.1 Event là gì? 41](#_Toc189810679)  [5.1.1.4.2.2 Cách tạo và sử dụng event 41](#_Toc189810680)  [5.1.1.4.3 Tóm tắt nhanh 44](#_Toc189810681)  [5.1.1.4.4 Bài tập thực hành 44](#_Toc189810682) |

# Xu hướng game và vị trí tuyển dụng

## Game 2D:

**Ưu điểm**: Dễ học và phát triển hơn so với game 3D, phù hợp để xây dựng project đầu tay.

**Ứng dụng**: Game mobile (casual, puzzle) rất phổ biến.

**Tuyển dụng**: Nhiều công ty game mobile trong và ngoài nước yêu cầu Unity Developer với kỹ năng phát triển game 2D.

## Game 3D:

**Ưu điểm**: Được sử dụng trong game console và PC, đồ họa ấn tượng, cơ hội học nâng cao kỹ năng.

**Ứng dụng**: Game nhập vai, FPS, simulation.

**Tuyển dụng**: Các studio lớn như Ubisoft, Gameloft, VNG thường yêu cầu kỹ năng 3D với Unity.

# Xu hướng:

Game mobile đang chiếm ưu thế tại Việt Nam và quốc tế.

Game casual và hyper-casual 2D hoặc 3D được nhiều người chơi.

Kỹ năng làm multiplayer game hoặc tích hợp AI cũng rất được săn đón.

**Gợi ý**: Bắt đầu với **game 2D** để làm quen Unity, sau đó chuyển sang **game 3D** để mở rộng kỹ năng.

# Lộ trình học Unity trong 6 tháng

## Giai đoạn 1: Nền tảng cơ bản (1-1.5 tháng)

**Mục tiêu**: Làm quen Unity, hiểu lập trình C# cơ bản, và hoàn thành một mini-project game 2D.

**Công việc**:

### Học C# cơ bản và ứng dụng trong Unity:

* + - Các khái niệm: biến, hàm, vòng lặp, mảng, đối tượng (OOP cơ bản).
    - Tài liệu: C# Yellow Book (miễn phí) hoặc khóa học online như [Learn C# by Brackeys](https://www.youtube.com/@Brackeys).

### Làm quen Unity:

* + - Tải và cài đặt Unity Editor (bản Personal miễn phí).
    - Hiểu giao diện Unity (Scene, Game, Hierarchy, Inspector).
    - Tài liệu: Unity Learn (miễn phí).

### Thực hành mini-project:

* + - Làm game 2D cơ bản như Flappy Bird hoặc Pong.

## Giai đoạn 2: Phát triển kỹ năng nâng cao (1.5-2 tháng)

**Mục tiêu**: Hiểu sâu hơn về Unity và hoàn thiện một game 2D chất lượng.

**Công việc**:

### Học kỹ năng nâng cao:

* + - Vật lý (Physics2D): Collider, RigidBody.
    - Hệ thống Animation, UI, Sound.
    - ScriptableObject và quản lý dữ liệu.

### Làm game 2D nâng cao:

* + - Ví dụ: Game platformer (Mario-like).
    - Tích hợp hệ thống lưu trữ, menu UI, và âm thanh.
    - Tài liệu: Khóa học như Complete 2D Game Developer Course (trả phí).

## Giai đoạn 3: Học làm game 3D (1.5-2 tháng)

* **Mục tiêu**: Làm quen với Unity 3D, nắm vững các kỹ thuật cơ bản.
* **Công việc**:
  1. **Làm quen với Unity 3D**:
     + Camera, ánh sáng, và môi trường 3D.
     + Xử lý vật lý 3D: Gravity, Colliders, và NavMesh.
  2. **Phát triển game 3D nhỏ**:
     + Ví dụ: Game bắn súng FPS đơn giản hoặc game đua xe.
     + Tài liệu: GameDev.tv's Unity 3D course (trả phí).

## Giai đoạn 4: Xây dựng project hoàn chỉnh (1-2 tháng)

* **Mục tiêu**: Phát triển một game hoàn chỉnh để làm portfolio.
* **Công việc**:
  1. **Chọn loại game**:
     + Nếu thích game 2D: Làm game puzzle hoặc RPG 2D.
     + Nếu thích game 3D: Làm game FPS hoặc platformer.
  2. **Hoàn thiện và tối ưu hóa**:
     + Thêm hiệu ứng âm thanh, hình ảnh.
     + Test và sửa lỗi.
     + Đóng gói sản phẩm và xuất bản lên các nền tảng như itch.io hoặc Google Play.

**Khóa học gợi ý trong ngân sách**

1. **Miễn phí**:
   * Unity Learn (learn.unity.com).
   * YouTube: Brackeys, Code Monkey, Dani Dev.
2. **Trả phí (dưới 10 triệu)**:
   * Complete Unity Developer 2D & 3D (Udemy, khoảng 500k-1 triệu).
   * [GameDev.tv Courses](https://www.gamedev.tv/).

**Cam kết và lưu ý**

* Học đều đặn mỗi ngày 2-3 giờ, cuối tuần dành thêm thời gian để thực hành project.
* Kết hợp lý thuyết với thực hành, hoàn thành từng mini-project trước khi chuyển sang phần tiếp theo.
* Tạo portfolio cá nhân (trang web hoặc GitHub) để trình bày các dự án của bạn.

# Chi tiết các giai đoạn học

## Giai đoạn 1: Nền tảng cơ bản (1-1.5 tháng)

**Mục tiêu**:

* Hiểu các khái niệm cơ bản trong lập trình C# (ngôn ngữ chính của Unity).
* Làm quen với giao diện và cách sử dụng Unity Editor.
* Phát triển một mini-project game 2D đơn giản.

### Tuần 1-2: Học C# cơ bản và ứng dụng trong Unity

#### Nội dung học:

* + **Cấu trúc cơ bản của C#:**
    - Biến và kiểu dữ liệu: int, float, string, bool.
    - Câu lệnh điều kiện: if, else, switch.
    - Vòng lặp: for, while, foreach.
  + **Hàm và lớp:**
    - Hàm (function): cách viết và sử dụng.
    - Khái niệm OOP: lớp (class), đối tượng, thuộc tính, và phương thức.
  + **Xử lý sự kiện (Events)**:
    - Hiểu cách liên kết các sự kiện với hành động trong game.

#### Nguồn học:

* + Miễn phí:
    - C# Yellow Book: Sách miễn phí giải thích C# chi tiết.
    - YouTube: [C# for Beginners - Brackeys](https://www.youtube.com/playlist?list=PLPV2KyIb3jR5Q2KzUuV4QfKpkThrMZG8c).
  + Trả phí (rẻ):
    - C# Programming for Unity Game Development trên Udemy (~500k khi giảm giá).

#### Thực hành:

* + Tạo một dự án C# độc lập (không cần Unity) để làm quen:
    - Bài tập: Tạo chương trình "đoán số" đơn giản.
    - Bài tập: Tạo ứng dụng quản lý danh sách công việc (to-do list).

### Tuần 3: Làm quen Unity Editor

#### Nội dung học:

* + **Cài đặt và thiết lập Unity**:
    - Tải và cài đặt Unity Hub.
    - Cài đặt Unity Editor (phiên bản LTS – Long Term Support).
  + **Giao diện Unity**:
    - Scene View, Game View, Hierarchy, Inspector, Project, Console.
  + **Các khái niệm cơ bản**:
    - GameObject và Component.
    - Transform: vị trí, xoay, và kích thước.
    - Prefab: tái sử dụng đối tượng.

#### Nguồn học:

* + Miễn phí:
    - Unity Learn: Unity Essentials Pathway.
    - YouTube: [Unity Beginner Tutorial - Brackeys](https://www.youtube.com/playlist?list=PLPV2KyIb3jRZHQ8WVHvDfblNpv0hsvjpS).
  + Trả phí:
    - Udemy: The Ultimate Beginner's Guide to Unity (~500k).

#### Thực hành:

* + Tạo một dự án Unity mới:
    - Thêm GameObject (Cube, Sphere) vào Scene.
    - Tạo Script C# để điều chỉnh vị trí của GameObject (dùng transform.Translate).

### Tuần 4-5: Mini-project game 2D

#### Mục tiêu:

* + Hoàn thiện một game 2D đơn giản để áp dụng kiến thức học được.

#### Dự án mẫu: Flappy Bird Clone

* + **Yêu cầu**:
    - Người chơi điều khiển một chú chim, tránh va chạm vào chướng ngại vật.
  + **Các bước thực hiện**:
    - **Thiết lập game**:
      * Thêm nền (background) và đối tượng (bird, pipes).
      * Sử dụng Sprite Renderer để hiển thị hình ảnh.
    - **Viết script**:
      * Điều khiển chú chim (nhảy lên khi nhấn phím).
      * Di chuyển chướng ngại vật (pipes) qua màn hình.
    - **Thêm tính năng**:
      * Hệ thống điểm: Tăng điểm khi vượt qua chướng ngại vật.
      * Reset game khi va chạm.
    - **Hoàn thiện**:
      * Thêm âm thanh (nhảy, va chạm).
      * Làm UI đơn giản (menu bắt đầu, hiển thị điểm).

#### Nguồn học:

* + Miễn phí:
    - [Brackeys - Make a Flappy Bird in Unity](https://www.youtube.com/watch?v=bz7KntmjQ2Y).
  + Trả phí:
    - Udemy: Build a 2D Unity Game (~500k).

#### Lưu ý

* **Thời gian**:
  + Dành 1-1.5 giờ/ngày cho lý thuyết và video hướng dẫn.
  + Dành 1-1.5 giờ/ngày cho thực hành.

#### Tài liệu bổ sung:

* + Tham gia cộng đồng Unity (Discord, Reddit) để hỏi đáp.
  + Đọc tài liệu chính thức: Unity Documentation.

## Giai đoạn 2: Phát triển kỹ năng nâng cao (1.5-2 tháng)

**Mục tiêu**:

* Nắm vững các tính năng quan trọng trong Unity, bao gồm vật lý, animation, UI, và quản lý dữ liệu.
* Phát triển một game 2D nâng cao với các tính năng chuyên sâu, tối ưu hóa game để chuẩn bị cho dự án lớn hơn.

### Tuần 1-2: Hiểu sâu về Unity và C# nâng cao

**1. Kỹ thuật lập trình nâng cao trong Unity**

1. **ScriptableObject**:
   * **Công dụng**: Lưu trữ dữ liệu cố định như bảng thông số, danh sách item.
   * **Học qua ví dụ**:
     + Tạo một ScriptableObject để lưu dữ liệu về các nhân vật (máu, tốc độ, sát thương).
   * **Nguồn học**:
     + ScriptableObject Basics - Unity Learn.
2. **Quản lý sự kiện (Event System)**:
   * **Công dụng**: Tăng tính modular cho game bằng cách dùng sự kiện thay vì liên kết trực tiếp giữa các object.
   * **Thực hành**:
     + Tạo hệ thống điểm số tự động cập nhật khi người chơi đạt được thành tích.
   * **Tài liệu**:
     + Unity Event System Documentation.
3. **Tổ chức mã tốt hơn**:
   * **Tips**:
     + Sử dụng các lớp riêng biệt để quản lý logic game, âm thanh, UI.
     + Áp dụng Design Pattern: Singleton (quản lý trạng thái chung).

**2. Xử lý vật lý 2D nâng cao**

1. **Các thành phần vật lý 2D chính**:
   * RigidBody2D: Kích hoạt và điều chỉnh trọng lực, lực đẩy.
   * Collider2D: Hình dạng va chạm (Box, Circle, Polygon).
   * Physics Material: Tạo hiệu ứng ma sát, đàn hồi.
2. **Thực hành**:
   * Tạo hiệu ứng nảy bóng (sử dụng Physics Material).
   * Xây dựng hệ thống va chạm giữa nhân vật và môi trường (platformer).

**3. Hệ thống Animation:**

1. **Animator vs Animation Clip**:
   * Tạo animation clip cơ bản (chuyển động nhảy, chạy).
   * Sử dụng Animator Controller để chuyển đổi trạng thái (State Machine).
2. **Thực hành**:
   * Tạo nhân vật với hoạt ảnh đi, nhảy, đứng yên.
   * Thêm logic chuyển trạng thái dựa trên input người chơi.

### Tuần 3-4: Làm game 2D nâng cao

**Dự án mẫu: Game platformer (Mario-like)**

**Mô tả**:  
Người chơi điều khiển nhân vật nhảy qua các platform, thu thập coin, tránh chướng ngại vật, và về đích.

**Bước 1: Thiết lập cơ bản**

1. **Bố trí môi trường**:
   * Tạo các đối tượng nền, platform, và chướng ngại vật.
   * Sử dụng Tilemap (Unity 2D) để dễ dàng vẽ và bố trí level.
2. **Nhân vật chính**:
   * Thêm Sprite nhân vật.
   * Tạo script điều khiển di chuyển (chạy, nhảy) và tương tác với môi trường.

**Bước 2: Tính năng chính**

1. **Vật lý và cơ chế game**:
   * Hệ thống trọng lực: Dùng RigidBody2D để nhảy và rơi.
   * Phát hiện va chạm: Nhân vật mất máu khi chạm vào chướng ngại vật.
2. **Hệ thống thu thập**:
   * Thu thập coin: Tăng điểm khi nhặt coin.
   * Hiển thị điểm trên màn hình (sử dụng UI Canvas).
3. **AI cơ bản**:
   * Tạo kẻ thù di chuyển qua lại.
   * Script hành vi của kẻ thù: Tấn công nhân vật khi lại gần.

**Bước 3: Hoàn thiện và tối ưu hóa**

1. **UI và menu**:
   * Tạo màn hình bắt đầu, màn hình thắng/thua.
   * Thêm thanh máu (health bar) cho nhân vật.
2. **Hiệu ứng**:
   * Âm thanh: Nhạc nền, âm thanh khi nhảy, va chạm.
   * Particles: Tạo hiệu ứng bụi khi nhân vật nhảy.
3. **Level design**:
   * Tạo thêm 2-3 level với độ khó tăng dần.
4. **Test và tối ưu**:
   * Test trên nhiều độ phân giải và cấu hình.
   * Tối ưu FPS (frames per second)

**Nguồn học**

1. **Miễn phí**:
   * [Make a 2D Platformer Game](https://www.youtube.com/watch?v=RY6nx0rELTc) - GameMaker's Toolkit.
   * Unity Tilemap Tutorial - Unity Learn.
2. **Trả phí**:
   * Udemy: Complete 2D Game Developer (~500k).
3. **Thời gian biểu đề xuất**

| **Ngày** | **Hoạt động** | **Thời gian** |
| --- | --- | --- |
| Ngày 1-7 | Học vật lý 2D và thực hành Physics Material, Colliders. | 2-3 giờ/ngày |
| Ngày 8-14 | Học Animation, tạo các trạng thái di chuyển của nhân vật. | 2-3 giờ/ngày |
| Ngày 15-21 | Thiết lập game platformer cơ bản (nhân vật, môi trường). | 3-4 giờ/ngày |
| Ngày 22-28 | Thêm AI, UI, hiệu ứng và hoàn thiện game. | 3-4 giờ/ngày |

**Kết quả cuối cùng**

* Bạn sẽ có một game platformer 2D hoàn chỉnh với các tính năng cơ bản và nâng cao.
* Đây sẽ là sản phẩm nổi bật để thêm vào portfolio hoặc làm nền tảng để mở rộng hơn nữa.

## Giai đoạn 3: Học làm game 3D (1.5-2 tháng)

Mục tiêu:

* Làm quen với các công cụ và kỹ thuật phát triển game 3D trong Unity.
* Tạo một game 3D cơ bản để thực hành kỹ năng.

### Tuần 1: Làm quen với Unity 3D

1. Khái niệm cơ bản trong Unity 3D

1. Scene trong không gian 3D:
   * Trục tọa độ 3D (X, Y, Z).
   * Camera và ánh sáng trong môi trường 3D.
2. GameObject trong 3D:
   * Mesh (hình học 3D cơ bản như Cube, Sphere).
   * Collider 3D (Box, Sphere, Capsule).
   * RigidBody (vật lý cơ bản).

2. Thực hành cơ bản

1. Tạo một scene 3D:
   * Thêm một mặt phẳng làm sàn, thêm Cube và Sphere.
   * Áp dụng vật liệu (Material) để thay đổi màu sắc và texture.
2. Scripting cơ bản trong 3D:
   * Tạo script để di chuyển một vật thể (như Cube) bằng bàn phím.

3. Nguồn học

* Unity Learn - 3D Beginner Tutorials.
* [3D Game Development Basics - YouTube](https://www.youtube.com/watch?v=on9nwbZngyw).

### Tuần 2: Làm việc với camera và ánh sáng

1. Camera trong Unity 3D

1. Loại camera:
   * Fixed camera (góc nhìn cố định).
   * Follow camera (camera theo sát nhân vật).
   * First-person và third-person camera.
2. Thực hành:
   * Tạo một camera đi theo nhân vật trong không gian 3D.

2. Ánh sáng và hiệu ứng bóng (Lighting)

1. Loại ánh sáng:
   * Directional Light (ánh sáng mặt trời).
   * Point Light (nguồn sáng điểm).
   * Spot Light (ánh sáng chiếu từ góc cố định).
2. Thực hành:
   * Thêm ánh sáng mặt trời vào scene.
   * Tạo bóng động (shadow) cho các vật thể.

Nguồn học

* Lighting in Unity - Official Tutorials.

### Tuần 3-4: Tạo game 3D cơ bản (Dự án nhỏ)

**Ý tưởng game:**

**Roll-a-ball**: Người chơi điều khiển một quả bóng di chuyển trên một mặt phẳng, thu thập các item và tránh chướng ngại vật.

**1. Thiết lập cơ bản**

1. **Môi trường**:
   * Tạo một mặt phẳng lớn làm sàn.
   * Thêm các vật thể làm chướng ngại vật (Cube, Sphere).
2. **Nhân vật**:
   * Thêm một quả bóng (Sphere) làm nhân vật chính.
   * Áp dụng RigidBody để điều khiển bóng bằng lực (Physics).

**2. Tính năng cơ bản**

1. **Điều khiển bóng**:
   * Viết script để bóng di chuyển bằng các phím mũi tên hoặc WASD.
2. **Thu thập item**:
   * Thêm các Cube nhỏ làm item.
   * Viết script để bóng biến mất khi chạm vào item (sử dụng trigger collider).

**3. Tính năng nâng cao**

1. **UI**:
   * Hiển thị điểm số của người chơi.
   * Tạo màn hình thắng/thua.
2. **Camera**:
   * Tạo camera theo sau bóng (smooth follow).

**Nguồn học**

* Roll-a-ball Tutorial - Unity Learn.
* [Complete 3D Game in Unity](https://www.youtube.com/watch?v=5CgWbhqjUsM).

### Tuần 5-6: Làm game 3D nâng cao (Dự án thứ hai)

**Ý tưởng game:**

**Third-Person Shooter**: Người chơi điều khiển một nhân vật, di chuyển qua các khu vực và bắn kẻ thù.

**1. Thiết lập môi trường**

1. **Nhân vật**:
   * Thêm mô hình 3D (tải từ Unity Asset Store hoặc sử dụng mô hình placeholder).
   * Tạo hoạt ảnh (animation) cho nhân vật: đi, chạy, nhảy.
2. **Camera và điều khiển nhân vật**:
   * Sử dụng Third-Person Controller để điều khiển nhân vật.
   * Camera theo sát nhân vật từ góc nhìn thứ ba.

**2. Tính năng cơ bản**

1. **Hệ thống bắn súng**:
   * Viết script để bắn đạn (Projectile).
   * Thêm hiệu ứng khi bắn trúng kẻ thù (Particle Effect).
2. **Hệ thống AI đơn giản**:
   * Kẻ thù tuần tra khu vực.
   * Kẻ thù truy đuổi và tấn công người chơi khi đến gần.

**3. Hoàn thiện**

1. **Level Design**:
   * Tạo khu vực chơi với chướng ngại vật và item.
2. **Hiệu ứng và tối ưu hóa**:
   * Thêm ánh sáng, texture, và nhạc nền.
   * Tối ưu FPS.

**Nguồn học**

* [Third-Person Controller - Brackeys](https://www.youtube.com/watch?v=4HpC--2iowE).
* Unity Shooter Game Tutorial.

**Kết quả cuối cùng**

* Bạn sẽ có ít nhất 2 game 3D (1 game đơn giản như Roll-a-ball và 1 game phức tạp hơn như Third-Person Shooter).
* Kỹ năng Unity 3D của bạn sẽ đủ để chuyển sang giai đoạn 4: phát triển dự án hoàn chỉnh.

## Giai đoạn 4: Xây dựng project hoàn chỉnh (1-2 tháng)

**Mục tiêu**:

* Xây dựng một game hoàn chỉnh (2D hoặc 3D) từ ý tưởng đến sản phẩm cuối cùng.
* Game có đầy đủ tính năng, tối ưu hóa tốt và thể hiện rõ kỹ năng bạn đã học.
* Sản phẩm này sẽ là dự án chính để đưa vào portfolio cá nhân.

### Tuần 1: Lên ý tưởng và lập kế hoạch chi tiết

**1. Chọn ý tưởng game**

Bạn nên chọn một ý tưởng game có thể hoàn thành trong thời gian giới hạn và phù hợp với vị trí bạn muốn ứng tuyển. Một số gợi ý:

1. **Game 2D**:
   * **Platformer nâng cao** (có hệ thống level, bẫy, kẻ thù, vật phẩm).
   * **Puzzle game** với cơ chế độc đáo.
2. **Game 3D**:
   * **Third-Person Adventure Game**: Người chơi điều khiển nhân vật đi qua các khu vực, giải đố và chiến đấu.
   * **First-Person Shooter** hoặc game đua xe cơ bản.

**2. Xác định tính năng chính (MVP)**

* Lập danh sách các tính năng cần có:
  + **Gameplay chính**: Cách chơi chính (di chuyển, chiến đấu, giải đố).
  + **Giao diện người dùng (UI)**: Menu chính, thanh máu, điểm số.
  + **Mức độ khó**: Cấp độ hoặc khu vực tăng dần độ khó.
  + **Hiệu ứng**: Âm thanh, ánh sáng, particle.

**3. Lập kế hoạch phát triển**

* **Chia nhỏ nhiệm vụ**:
  + Giai đoạn 1: Thiết lập môi trường và gameplay cơ bản.
  + Giai đoạn 2: Phát triển tính năng nâng cao (AI, UI).
  + Giai đoạn 3: Hoàn thiện, testing, tối ưu hóa.
* **Công cụ hỗ trợ**:
  + **Trello** hoặc **Notion** để quản lý công việc.
  + **Git** để quản lý mã nguồn.

**4. Tìm tài nguyên**

* **Mô hình và texture**: Unity Asset Store, [Free3D](https://free3d.com/), [OpenGameArt](https://opengameart.org/).
* **Âm thanh**: [Freesound](https://freesound.org/), [Zapsplat](https://www.zapsplat.com/).

### Tuần 2-3: Phát triển dự án

**1. Xây dựng gameplay chính**

1. **Thiết lập môi trường**:
   * Tạo khu vực chơi (level đầu tiên).
   * Sử dụng các asset 2D hoặc 3D để xây dựng cảnh quan.
2. **Gameplay cơ bản**:
   * Nhân vật di chuyển, nhảy (nếu là platformer).
   * Hệ thống bắn súng, đánh cận chiến (nếu là game chiến đấu).
3. **Tương tác vật lý**:
   * Va chạm giữa nhân vật và môi trường.
   * Các vật phẩm như tiền, vũ khí, hoặc cổng dịch chuyển.

**2. Tích hợp tính năng nâng cao**

1. **AI kẻ thù**:
   * Hành vi đơn giản: tuần tra, truy đuổi, tấn công.
   * Hành vi phức tạp: né tránh, phối hợp (nếu có thời gian).
2. **UI và HUD**:
   * Thanh máu, thanh năng lượng.
   * Hiển thị điểm số hoặc trạng thái nhiệm vụ.
3. **Camera và ánh sáng**:
   * Camera động (như third-person hoặc follow cam).
   * Hiệu ứng ánh sáng theo bối cảnh.

### Tuần 4-5: Hoàn thiện và tối ưu hóa

**1. Testing và sửa lỗi**

* **Kiểm tra logic**:
  + Tính năng hoạt động như dự kiến không?
  + Có lỗi nào ảnh hưởng đến trải nghiệm không?
* **Kiểm tra hiệu suất**:
  + FPS có ổn định không?
  + Sử dụng Unity Profiler để tìm và khắc phục điểm nghẽn.

**2. Thêm yếu tố hoàn thiện**

* **Hiệu ứng đặc biệt**:
  + Particles (cháy nổ, bụi, ánh sáng).
  + Shader đơn giản để tăng độ thẩm mỹ.
* **Âm thanh**:
  + Hiệu ứng âm thanh khi va chạm, chiến đấu.
  + Nhạc nền phù hợp với bối cảnh.

**3. Xây dựng level hoàn chỉnh**

* Tạo từ 2-3 level với độ khó tăng dần.
* Mỗi level nên có yếu tố đặc trưng để tạo sự mới mẻ.

### Tuần 6: Chuẩn bị phát hành và tạo portfolio

**1. Đóng gói dự án**

* Tạo file build chạy trên PC (hoặc APK cho mobile).
* Đặt tên game, thiết kế biểu tượng và màn hình giới thiệu.

**2. Đăng tải dự án**

* **Itch.io**: Nền tảng phổ biến để chia sẻ game.
* **GitHub**: Chia sẻ mã nguồn (nếu cần).
* **YouTube**: Quay video demo gameplay.

**3. Cập nhật portfolio cá nhân**

* Mô tả dự án chi tiết:
  + Ý tưởng game.
  + Công nghệ đã sử dụng.
  + Hình ảnh, video demo.

**Kết quả cuối cùng**

* Một game hoàn chỉnh, thể hiện kỹ năng từ cơ bản đến nâng cao.
* Portfolio chuyên nghiệp sẵn sàng để ứng tuyển vào các công ty game trong và ngoài nước.

## Giai đoạn 5: Dự án hoàn chỉnh và chuẩn bị cho tuyển dụng (1.5-2.5 tháng)

**Mục tiêu**:

* Hoàn thiện một dự án game chuyên sâu, áp dụng các kỹ thuật nâng cao.
* Tạo portfolio cá nhân chuyên nghiệp để chuẩn bị cho việc ứng tuyển.

### Tuần 1-2: Lập kế hoạch và triển khai dự án hoàn chỉnh

**1. Chọn loại game và ý tưởng**

Bạn cần chọn một thể loại game phù hợp để gây ấn tượng với nhà tuyển dụng. Dưới đây là các gợi ý phổ biến:

1. **Game 2D**:
   * **Platformer nâng cao** (ví dụ: có hệ thống vũ khí, level đa dạng).
   * **Puzzle game** (ví dụ: giải đố với cơ chế vật lý).
2. **Game 3D** (nếu muốn thử thách):
   * **Game bắn súng góc nhìn thứ nhất (FPS)**.
   * **Game nhập vai (RPG)** đơn giản.

**2. Lập kế hoạch dự án**

1. **Định nghĩa tính năng**:
   * Lên danh sách các tính năng cần có (ví dụ: di chuyển nhân vật, AI kẻ thù, hệ thống level).
   * Phân loại tính năng thành cơ bản (MVP) và nâng cao (tính năng phụ).
2. **Công cụ hỗ trợ**:
   * **Trello/Notion**: Quản lý tiến độ dự án.
   * **Git**: Theo dõi và quản lý phiên bản mã nguồn.

**3. Tạo bản demo nhanh (prototype)**

* Tập trung vào các cơ chế chính (core mechanics).
* Chỉ cần làm các đối tượng cơ bản (placeholder), không cần đồ họa đẹp.

### Tuần 3-4: Phát triển và hoàn thiện dự án

**1. Các bước phát triển**

1. **Xây dựng core mechanics**:
   * Ví dụ: Nếu làm platformer, hoàn thiện hệ thống di chuyển, nhảy, và va chạm.
2. **Thêm tính năng nâng cao**:
   * **Hệ thống AI**:
     + Kẻ thù có hành vi phức tạp (tuần tra, truy đuổi, tấn công).
   * **Quản lý tài nguyên**:
     + Hệ thống vật phẩm (item), vũ khí, hoặc nâng cấp.
   * **UI nâng cao**:
     + Hệ thống menu chi tiết (chọn level, tùy chỉnh cài đặt).
3. **Hiệu ứng và tối ưu hóa**:
   * **Particles và Shader**: Tạo hiệu ứng đặc biệt (cháy nổ, ánh sáng).
   * **Âm thanh**: Tích hợp nhạc nền, hiệu ứng âm thanh.

**2. Testing và fix lỗi**

* **Chạy thử nghiệm trên các thiết bị khác nhau** (PC, mobile, nếu có).
* Mời bạn bè hoặc tham gia cộng đồng để nhận phản hồi

### Tuần 5-6: Hoàn thiện dự án và tạo portfolio

**1. Hoàn thiện dự án**

* **Thêm yếu tố thẩm mỹ**:
  + Tối ưu giao diện (UI/UX).
  + Sử dụng asset miễn phí hoặc trả phí từ Unity Asset Store để nâng cao đồ họa.
* **Tạo phiên bản hoàn chỉnh**:
  + Tạo file build (game chạy độc lập) cho PC hoặc APK cho mobile.

**2. Tạo portfolio cá nhân**

1. **Trình bày dự án**:
   * Viết mô tả chi tiết:
     + Giới thiệu ý tưởng game.
     + Các tính năng chính.
     + Công nghệ đã sử dụng (Unity, C#, AI, UI, v.v.).
   * Thêm hình ảnh và video demo.
2. **Công cụ tạo portfolio**:
   * Miễn phí: [GitHub Pages](https://pages.github.com/), [Notion](https://www.notion.so/).
   * Chuyên nghiệp: [Wix](https://www.wix.com/), [Squarespace](https://www.squarespace.com/).

**3. Đăng tải sản phẩm**

* Đăng lên các nền tảng như [Itch.io](https://itch.io/), [GitHub](https://github.com/), hoặc YouTube để tăng cơ hội tiếp cận nhà tuyển dụng.

**Nguồn học**

1. **Học nâng cao**:
   * Miễn phí: Unity Advanced Tutorials - Unity Learn.
   * Trả phí:
     + Udemy: Advanced Unity Game Development (~500k).
2. **Học hiệu ứng (VFX)**:
   * Miễn phí: [Making Particles in Unity - Brackeys](https://www.youtube.com/watch?v=llmK22bTgpY).
3. **Học tối ưu hóa**:
   * Optimize Your Game - Unity.

**Kết quả cuối cùng**

* Một dự án game hoàn chỉnh với đồ họa đẹp, tính năng phong phú, và tối ưu hóa tốt.
* Portfolio cá nhân ấn tượng, sẵn sàng để ứng tuyển.

# Nội dung bài học

## Giai đoạn 1: Nền tảng cơ bản

### Tuần 1 -2: Học C# cơ bản và ứng dụng trong Unity

#### Cấu trúc cơ bản của C#

* + 1. Biến và kiểu dữ liệu

Biến là một vùng nhớ dùng để lưu giá trị. C# yêu cầu bạn khai báo biến với kiểu dữ liệu cụ thể.

Kiểu dữ liệu cơ bản trong C#:

* + int: số nguyên
  + float: số thực. VD: float height = 1.75f ( có thêm hậu tố f để chỉ định kiểu float
  + string: chuỗi ký tự
  + bool: kiểu logic, true false
    1. Câu lệnh điều kiện

Câu lệnh điều kiện cho phép chương tình thực hiện các hành động khác nhau dựa trên điều kiện logic

* + Cấu trúc if – else

A computer screen with text on it

Description automatically generated

* + Cấu trúc switch: kiểm tra giá trị cụ thể của biến

A computer screen with white and blue text

Description automatically generated

* + 1. Vòng lặp

Giúp thực hiện lặp đi lặp lại một đoạn mã nhiều lần

3.1. Vòng lặp for: sử dụng khi bạn biết trước số lần lặp

A black screen with white text

Description automatically generated

3.2. Vòng lặp while: sử dụng khi điều kiện xác định việc tiếp tục vòng lặp ( không biết trước số vòng lặp )

A computer screen with white text

Description automatically generated

3.3. Vòng lặp foreach: Dùng để duyệt qua các phần tử trong một mảng or danh sách

A screenshot of a computer screen

Description automatically generated

#### Hàm và lớp ( OPP cơ bản)

**Lý thuyết**

Hàm ( function ): là một đoạn mã có thể tái sử dụng để thực hiện một nhiệm vụ cụ thể

**Cách khai báo**

A computer screen with white text

Description automatically generated

VD

A computer screen shot of a program code

Description automatically generated

#### Khái niệm OOP ( Lập trình hướng đối tượng )

* 1. **Lớp ( Class )**
  + Khuôn mẫu để tạo ra các đối tượng.
  + Bao gồm thuộc tính ( dữ liệu ) và phương thức ( hành vi)
  1. **Đối tượng ( Object)**
  + Thể hiện của một lớp
  1. **Cú pháp**

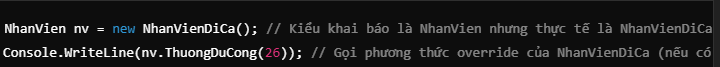
A screen shot of a computer program

Description automatically generated

* 1. **Các tính chất**
  2. **Tính đa hình – kế thừa**
     1. **Từ khóa virtual và orverride**
  + Virtual: được sử dụng trong lớp cha để cho phép các lớp con **ghi đè**  ( override ) phương thức đó. Sử dụng tử khóa override trong lớp con, phương thức của lớp con sẽ thay thế hoàn toàn phương thức của lớp cha **khi gọi qua đối tượng kiểu lớp cha.**
  + Khi nào nên dùng virtual và override

+ Khi muốn các lớp con **tùy chỉnh hành vi của phương thức,** nhưng vẫn giữ khả năng gọi phương thức đó từ lớp cha

+ Override đảm bảo tính đa hình ( Polymorphism): Hành vi của đối tượng sẽ được xác định dựa trên kiểu thực tế ( runtime)



Trong trường hợp này

* + Nếu phương thức ThuongDuCong sử dụng virtual trong lớp cha ( NhanVien) và override trong lớp con ( NhanVienDiCa) phương thức ghi đè sẽ được gọi ( phương thức trong lớp con).
    1. **Từ khóa new**
  + New: được sử dụng trong lớp con để **che giấu ( hide )** phương thúc của lớp cha. Phương thức lớp con sẽ không ghi đè phương thức lớp cha mà chỉ thay thế nó khi **gọi qua đối tượng kiểu lớp con.**
  + Khi nào nên dùng new?

+ Khi bạn không muốn/không cần đa hình

+ Khi bạn chỉ muốn định nghĩa một phương thức mới, độc lập, mà không ảnh hưởng đến phương thức của lớp cha.

A screenshot of a computer

Description automatically generated

Trong trường hợp này

+ Phương thức TinhLuong của lớp cha sẽ được gọi khi kiểu khai báo là NhanVien

+Phương thức TinhLuong của lớp con sẽ được gọi khi kiểu khai báo là NhanVienDiCa

* + 1. **Tóm lại**
  + Sử dụng virtual và override khi cần tính đa hình, lớp con ghi đè ( thay thế ) hoàn toàn phương thức của lớp cha.
  + Sử dụng new: gọi kết quả từ method của lớp cha và sử dụng kết quả đó để tính ở method ở lớp con.

1. **Xử lý sự kiện ( Events):** Hiểu cách liên kết các sự kiện với hành động trong game

#### Delegate và Event

**Mục tiêu**

**Event và Delegate trong C#**

Delegate là gì? Cách khai báo và sử dụng

Event là gì? Cách tạo và đăng ký sự kiện

Ví dụ cơ bản về sự kiện

**Ứng dụng Event trong Unity**

Bắt sự kiện từ bàn phím và chuột

Xử lý va chạm ( Collision Event).

Unity Event System ( Onclick, OntriggerEnter…)

**Bài tập thực hành (không cần Unity)**

1. Viết chương trình mô phỏng sự kiện bấm nút:
   * Khi người dùng nhập "1", hiển thị "Bạn đã bấm nút 1!".
   * Khi nhập "2", hiển thị "Bạn đã bấm nút 2!".
2. Viết chương trình mô phỏng đồng hồ:
   * Mỗi giây, chương trình tự động thông báo "Đã trôi qua 1 giây".

**Nội dung kiến thức**

##### Delegate là gì?

Delegate trong C# là một kiểu dữ liệu đặc biệt, cho phép lưu trữ tham chiếu đến một hoặc nhiều phương thức. Bạn có thể hiểu delegate giống như một “con trỏ hàm” trong C/C++.

Delegate giúp chúng ta gọi các phương thức gián tiếp, có thể thay đổi phương thức được gọi mà không cần sửa mã nguồn.

VD: nếu bạn có một chức năng thông báo ( Notify), bạn có thể sử dụng delegate để gọi các phương thức khác nhau như gửi email, hiển thị thông báo trên màn hình, hoặc ghi log.

###### Cách khai báo và sử dụng delegate

**Bước 1**: Khai báo delegate

Delegate được khai báo bằng từ khóa delegate, có cú pháp

A black and white text

AI-generated content may be incorrect.

VD

A black and white screen with white text

AI-generated content may be incorrect.

**Bước 2**: Gán phương thức cho delegate

Sau khi khai báo delegate, chúng ta có thể tạo 1 biến delegate và gắn một phương thức có cùng kiểu với delegate

A screen shot of a computer program

AI-generated content may be incorrect.

Kết quả:

A black and grey rectangular object

AI-generated content may be incorrect.

**Bước 3:** Gán nhiều phương thức cho một delegate

Một delegate có thể tham chiếu nhiều phương thức cùng lúc bằng cách sử dụng toán tử += ( Multicast Delegate).

A screenshot of a computer program

AI-generated content may be incorrect.

Kết quả:

A black rectangular object with a black rectangle

AI-generated content may be incorrect.

###### Xóa một phương thức khỏi delegate

Chúng ta có thể xóa một phưng thức khỏi delegate bằng cách sử dụng -=

A black and grey background

AI-generated content may be incorrect.

##### Event trong C#

###### Event là gì?

Event ( sự kiện) là một cách để giao tiếp giữa các đối tượng trong C#. Nó sử dụng delegate để thông báo rằng một sự kiện nào đó đã xảy ra.

Event giúp chúng ta thiết kế chương tình theo mô hình “Lắng nghe và Phản hồi” ( Observer Pattern).

VD: Khi người dùng bấm vào nút trên giao diện, một sự kiện “Click” sẽ được kích hoạt.

###### Cách tạo và sử dụng event

**Bước 1:** Khai báo delegate và event

* + Delegate dùng để định nghĩa kiểu phương thức sự kiện
  + Event được khai báo bằng từ khóa event dựa trên delegate

A screenshot of a computer program

AI-generated content may be incorrect.

**Kết quả**

A black and white screen

AI-generated content may be incorrect.

Một VD khác: Nút bấm trong Giao diện

Giả sử chúng ta có một nút bấm trong game, khi người chơi nhấn vào nút, chúng ta muốn hiển thị thông báo

A screen shot of a computer

AI-generated content may be incorrect.

A computer screen with text on it

AI-generated content may be incorrect.

**Kết quả**  
A black rectangular object with a black border

AI-generated content may be incorrect.

##### Tóm tắt nhanh

|  |  |
| --- | --- |
| Khái niệm | Mô tả |
| Delegate | Đại diện cho một hoặc nhiều phương thức có cùng kiểu. |
| Event | Cách giao tiếp giữa các đối tượng, dựa trên delegate |
| Multicast Delegate | Một delegate có thể trỏ đến nhiều phương thức cùng lúc. |
| Sử dụng event | Định nghĩa bằng event, đăng ký ( +=), hủy đăng ký (-=), và kích hoạt .Invoke(). |

##### Bài tập thực hành

1. Viết chương trình mô phỏng sự kiện bấm nút:
   * Khi người dùng nhập "1", hiển thị "Bạn đã bấm nút 1!".
   * Khi nhập "2", hiển thị "Bạn đã bấm nút 2!".
2. Viết chương trình mô phỏng đồng hồ:
   * Mỗi giây, chương trình tự động thông báo "Đã trôi qua 2 giây".
3. Viết chương trình mô phỏng nút bấm: Khi nhấn vào một nút, chương trình hiển thị "Nút đã được bấm!" và ghi log vào file "Log.txt".
4. Viết chương trình hẹn giờ: Cứ sau 3 giây, chương trình tự động hiển thị "Báo thức! Đã đến giờ học!".

#### Thực hành tổng kết tuần

* + Tạo một dự án C# độc lập (không cần Unity) để làm quen:
    - Bài tập: Tạo chương trình "đoán số" đơn giản.
    - Bài tập: Tạo ứng dụng quản lý danh sách công việc (to-do list).

### Tuần 3: Làm quen Unity Editor

#### Giao diện Unity Editor

### A computer screen shot of a computer AI-generated content may be incorrect.



* + Scene View (1): Khu vực làm việc chính để thao tác với đối tượng
  + Game View (2): Hiển thi kết quả khi chạy game
  + Hierarchy (3): Danh sách các đối tượng trong Scene
  + Inspector (4): Xem và chỉnh sửa thuộc tính của đối tượng
  + Project (5): Quản lý tài nguyên như hình ảnh, âm thanh, script
  + Console (6): Hiển thị thông báo, lỗi trong game

#### Các khái niệm cơ bản

* + GameObject: Mọi thứ trong Unity ( nhân vật, đồ vật, camera, đèn…) đều là GameObject
  + Component: Các thuộc tính của GameObject ( transform, Rigidbody, Collider,…)
  + Transform: Điều chỉnh vị trí, xoay, kích thước của GameObject.
  + Prefab: Tạo một mẫu đối tượng có thể tái sử dụng

##### GameObject trong Unity

###### GameObject là gì?

GameObject là đơn vị cơ bản nhất trong Unity, đại diện cho **mọi đối tượng trong game.**

* + Mỗi gameObject có thể là **một nhân vật, một cái cây, một viên gạch, ánh sáng, camera..v..v**
  + GameObject **không có hình dạng hay chức năng riêng biệt** cho đến khi bạn **gắn thêm component** cho nó.

###### Cấu trúc của GameObject

Mỗi GameObject gồm

* + Tên GameObject ( hiển thị trong Hirerarchy)
  + Transform ( thuộc tính mặc định, giúp định vị, xoay, thay đổi kích thước).
  + Component ( có thể thêm để mở rộng chức năng)

###### VD về gameObject

VD: Bạn muốn tạo một chiếc xe trong game

* + Ban đầu, nó chỉ là một GameObject rỗng ( không có gì đặc biệt )
  + Bạn thêm các Component như:

+ Mesh Renderer ( Hiển thị mô hình 3D của xe)

+ Rigidbody ( giúp xe có vật lý, có thể di chuyển )

+ Collider ( giúp xe va chạm tới vật thể khác )

* + - * Lúc này, GameObject “Chiếc xe” đã trở thành một đối tượng có thể nhìn thấy, có thể di chuyển, có thể va chạm

###### Cách tạo một GameObject trong Unity

**Có 2 cách chính**

Cách 1: Tạo GameObject bằng Unity Editor

* + 1. Trong **Hierarchy**, nhấn chuột phải 🡪 Create Empty ( tạo gameObject rỗng)
  + 2. Hoặc chọn GameObject 🡪 3D Object 🡪 Cube ( tạo một Cube làm ví dụ)

Cách 2: Tạo GameObject bằng C# script

A computer screen with white and red text

AI-generated content may be incorrect.

###### Tổng kết

* + GameObject là nền tảng của mọi thứ trong Unity
  + GameObject chỉ là một khung rỗng, cần gắn Component để hoạt động
  + Bạn có thể tạo GameObject bằng tay hoặc bằng mẫ C#

##### Component trong Unity

###### Component là gì

Component ( thành phần ) là các mô-dun giúp GameObject có tính năng cụ thể

+ GameObject bản thân nó chỉ là một khung rỗng, không có hình dạng, không thể di chuyển, không có va chạm

+ Muốn GameObject có khả năng đặc biệt, bạn phải thêm Component cho nó

+ Mọi thứ trong Unity đều **hoạt động** dựa trên Component

###### Một số Component quan trọng

|  |  |  |
| --- | --- | --- |
| Tên Component | Chức năng | VD |
| Transform | Điều chỉnh vị trí, xoay, kích thước | Dịch chuyển nhân vật, quay camera |
| Mesh Renderer | Hiển thị mô hình 3D | Giúp cube, Sphere hiện lên trong game |
| Rigidbody | Thêm vật lý ( trọng lực, lực tác động ) | Giúp nhân vật rơi xuống, đẩy được hộp |
| Collider | Tạo vùng va chạm | Giúp nhân vật không xuyên tường |
| Audio Source | Phát âm thanh | Phát nhạc nền, tiếng bước chân |
| Script ( C#) | Viết code điều khiển đối tượng | Giúp nhân vật di chuyển, AI hoạt động |

###### VD về Component

**VD 1**: Thêm Rigidbody để Cube rơi xuống

1.Trong Unity, Chọn GameObject 🡪 3D Object 🡪 Cube

2.Trong Inspector, nhấn Add Component 🡪 tìm Rigidbody 🡪 thêm vào

3. Nhấn Play, cube sẽ tự động rơi xuống vì có trọng lực

**VD 2:** Thêm Collider để Cube va chạm tới mặt đất

1.Tạo một **Plane** làm mặt đất ( GameObject 🡪 3D Object 🡪 Plane)

2.Đảm bảo Cube có **Box Collider** ( mặc định có sẵn )

3.Nhấn **Play**, Cube sẽ chạm đất và không xuyên qua.

**VD3**: viết script điều khiển GameObject

Tạo Component bằng C# Script để điều khiển GameObject

A screenshot of a computer program

AI-generated content may be incorrect.

Cách sử dụng:

1.Tạo GameObject 🡪 3D Object 🡪 Cube

2.Tạo **Script** **mới** tên **DieuKhienGameObject**

3.Kéo Script vào Cube trong **Inspector** ( hoặc nhấn “add component” để thêm )

4.Nhấn **Play**, bấm phím D, Cube sẽ di chuyển sang phải.

###### Tổng kết

* + Component là gì: 🡪 là những thành phần giúp GameObject có khả năng đặc biệt.
  + Có thể thêm Component bằng tay ( trong Unity ) hoặc bằng mã C#
  + Một gameObject có thể có nhiều Component cùng lúc.
  + **Component quan trọng**: Transform, Rigidbody, Colliderm Script,…

###### Bài tập thực hành

Cách tạo và mở file script trong Unity

**B1: Tạo script mới**

1.Trong Hierarchy, chọn cube

2.Vào Inspector, nhấn Add Component

3. Gõ “New Script” trong ô tìm kiếm

4.Đặt tên script, VD: CubeController

5. Nhấn Create and Add

**B2: Mở script để chỉnh sửa**

Sau khi tạo script, có 3 cách để mở

* + Cách 1: Bấm đúp vào script tỏng Inspector
  + Cách 2: Vào Project 🡪 Assets 🡪 Scripts, bấm đúp vào script
  + Cách 3: Dùng visual studio mở script từ ngoài Unity

**B3: Viết code điều khiển cube**

VD: di chuyển Cube sang trái/phải bằng phím mũi tên

A computer screen with text on it

AI-generated content may be incorrect.

Giải thích:

* + Input.GetAxis(“Horizontal”) nhận tín hiệu từ bàn phím <-- 🡪 hoặc A/D
  + Transform.Translate(moveX, 0, 0); Giúp Cube di chuyển theo trục X

**Bước 4: Chạy thử script**

1.Lưu script trong Visual studio

2.Quay lại Unity, đợi Unity biên dịch

3.Nhấn Play và thử di chuyển Cube bawgf phím mũi tên trái/phải

##### Transform trong Unity

###### Transform là gì?

* + Transform là **component quan trọng nhất** của mọi GameObject trong Unity
  + Quản lý vị trí ( Position), xoay ( Rotation), và kích thước ( Scale) của GameObject trong không gian 3D
  + Mọi GameObject đều có Transform mặc định, không thẻ xóa Component này

###### Các thuộc tính chính của Transform

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Chức năng | VD |
| Position ( vị trí ) | Xác định vị trí của GameObject trong thế giới | Đặt nhân vật tại (0,0,0) hoặc (5,2,-3) |
| Rotation (Xoay) | Xoay GameObject theo trục X, Y, Z | Xoay camera 45° để nhìn xuống |
| Scale ( kích thước) | Phóng to, thu nhỏ GameObject | Nhân đôi kích thước của Cube |

###### Cách thay đổi Transform

**Cách 1**: Thay đổi trong Unity Editor

1.Chọn GameObject (VD: Cube)

2.Trong Inspector, tìm Transform

3.Chỉnh các giá trị Position, Rotation, Scale theo ý muốn

VD:

* + Position ( 2,3,0) 🡪 Cube di chuyển đến ( 2,3,0)
  + Rotation ( 0,25,0) 🡪 Cube xoay 45° quay trục Y
  + Scale (2,2,2) 🡪 Cube to gấp đôi ban đầu

**Cách 2**: thay đổi bằng C# Script

Di chuyển GameObject sang phải mỗi frame

A computer screen with text

AI-generated content may be incorrect.

Giải thích:

* + Transform.position lấy vị trí hiện tại của GameObject’
  + += new Vector3(0.1f, 0.,0) 🡪 dịch sang phải mỗi frame

Xoay GameObject mỗi giây

A computer screen with white text

AI-generated content may be incorrect.

Giải thích:

* + Transform.Rotate(0, 90 \* Time.deltaTime, 0); 🡪 Xoay 90 độ/giây quanh trục Y

Thay đổi kích thước GameObject

A black background with white text

AI-generated content may be incorrect.

Giải thích

* + Transform.localScale = new Vector3(2,2,2); 🡪 Gấp đôi kích thước

###### Global vs. Local Transform

Unity có 2 hệ tọa độ quan trọng

* + Global ( thế giới): toàn bộ không gian game
  + Local ( Cục bộ ): Dựa vào GameObject cha

Di chuyển theo hệ tọa độ thế giới ( global)

A black background with blue and white text

AI-generated content may be incorrect.

Di chuyển theo hệ tọa độ cục bộ (local)

A black background with white text

AI-generated content may be incorrect.

* + - * Khi một GameObject nằm trong GameObject khác ( cha – con ), nó sẽ chịu ảnh hưởng của vị trí cha

###### Tổng kết

* + Transform là component giúp GameObject có vị trí, xoay và kích thước
  + Có thể thay đổi trực tiếp trong Unity hoặc qua C# Script
  + Global và Local Transform khác nhau khi có GameObject cha – con

##### Prefab trong Unity

###### Prefab là gì?

* + Prefab ( Pre-fabricated Object) là một mẫu GameObject có thể tái sử dụng
  + Dùng để tạo nhiều bản sao giống nhau của một đối tượng mà **không cần làm lại từ đầu**
  + Nếu thay đổi Prefab gốc, tất cả các bản sao ( Instance) cũng được cập nhật

###### Tại sao cần dùng Prefab?

* + Tiết kiệm thời gian: tạo một lần, dùng nhiều lần
  + Dễ dàng quản lý: thay đổi prefab một lần, tất cả bản sao đều thay đổi theo
  + Tăng hiệu suất: prefab giúp quản lý tài nguyên tốt hơn, đặc biệt khi có nhiều đối tượng giống nhau (VD: NPC, đạn, kẻ địch…)

###### Cách tạo và sử dụng Prefab

**Bước 1: tạo một Prefab**

1.Tạo GameObject: Vào GameObject 🡪 3D Object 🡪 Cube ( hoặc Sphere, Capsule)

2.Thêm Component: Thêm Rigidbody, Collider,.. nếu cần

3.Kéo GameObject vào thư mục “Assets” 🡪 Unity sẽ tự động tạo một Prefab file

**Bước 2: Tạo bản sao từ Prefab ( Instance)**

Kéo Prefab từ **Assets** vào **Hierarchy** hoặc **Scene** để tạo một bản sao

**Bước 3: thay đổi Prefab và cập nhật tất cả bản sao**

1.Chọn Prefab gốc trong Assets

2.Chỉnh sửa các thuộc tính ( màu sắc, kích thước, script,..)

3.Lưu lại, tất cả các bản sao sẽ được cập nhật tự động

###### Tạo prefab bằng C# Script

VD: Tạo ra kẻ địch mỗi giây

A computer screen shot of a black screen

AI-generated content may be incorrect.

Giải thích

* + Instantiat(enemyPrefab, spawnpoint.position, Quaternion.identity); 🡪 tạo 1 bản sao từ Prefab tại vị trí spawnPoint.
  + InvokeRepeating (“SpawnEnemy”, 1f, 1f); 🡪 Gọi hàm SpawnEnemy() mỗi giây
  + Ứng dụng thực tế: sinh kẻ địch, đạn, vật phẩm rơi, hiệu ứng…

###### Prefab Variants ( Biến thể của Prefab)

Unity cho phép tạo Prefab Variants, tức là bạn có thể tạo **Prefab mới dựa trên một Prefab cũ**, nhưng vẫn giữ mỗi liên kết với Prefab gốc

VD:

* + Tạo một Prefab nhân vật cơ bản
  + Tạo một Prefab Variant cho nhân vật mạnh hơn ( tăng HP, thêm hiệu ứng)

Cách làm

1.Chuột phải vào Prefab gốc 🡪 Create Prefab Variant

2.Thay đổi các thuộc tính mong muốn mà không ảnh hưởng đến Prefab gốc

###### Tổng kết

* + Prefab giúp tái sử dụng GameObject một cách hiệu quả
  + Có thể tạo Prefab trong Editor/C# Script
  + Sử dụng Prefab để sinh ra nhiều bản sao giống nhau trong game ( kẻ địch, đạn, hiệu ứng…)
  + Prefab Variants giúp tùy chỉnh mà vẫn giữ liên kết với Prefab gốc.