# Bài 2: Sử dụng GUI thiết yếu, sử dụng các thành phần của GUI và Script Communication

*Giảng viên:*

# Mục tiêu

- Các thành phần của GUI
- GUI Layout
- Sử dụng Styles và Skins cho Design ‘Look and Feel’
- Script cho GUI và tương tác với đối tượng trong thế giới game

# Các thành phần GUI

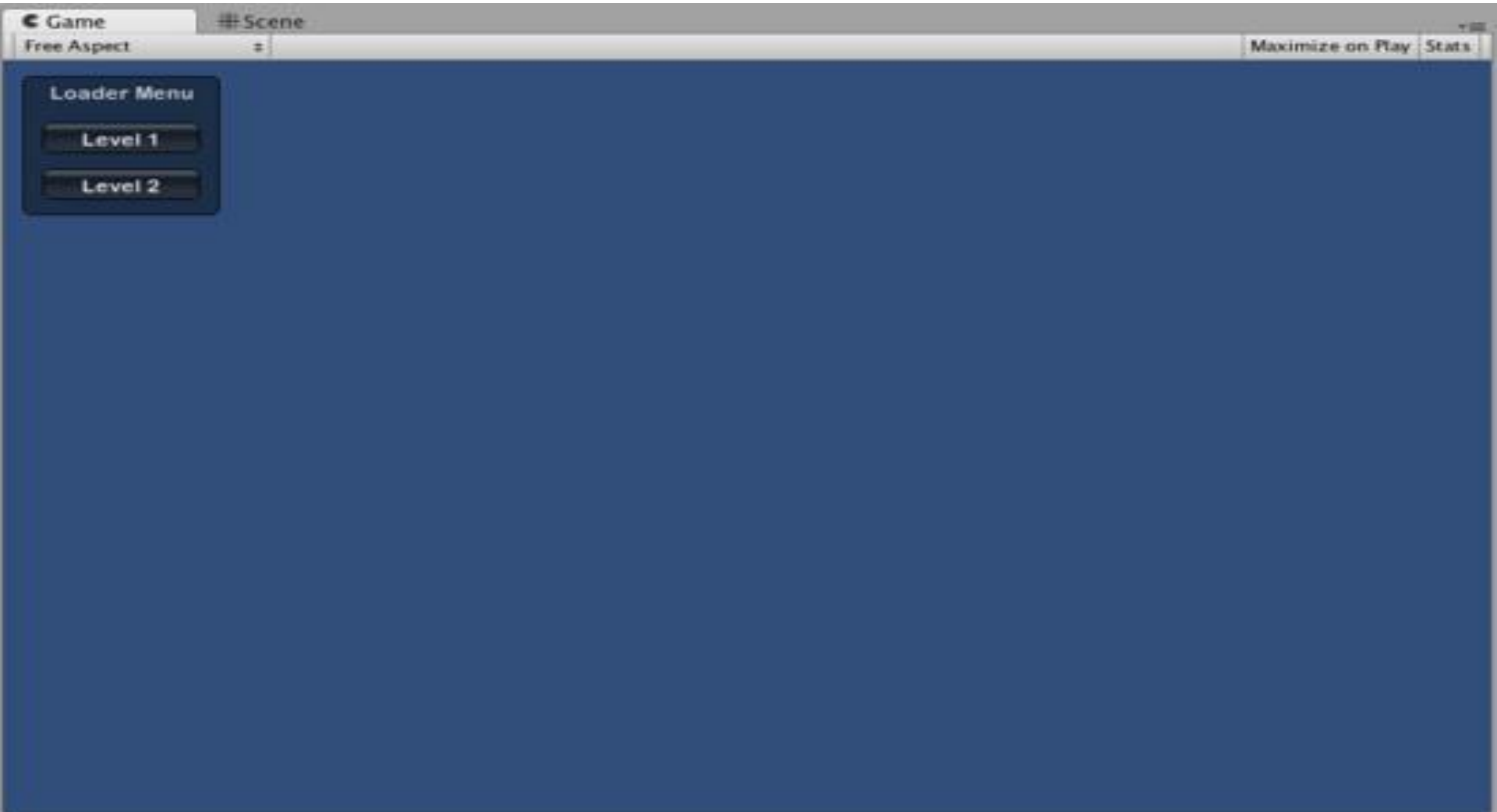
- Graphical User Interface, viết tắt là GUI. Thư viện thành phần chứa tất cả các thành tố phục vụ cho việc xây dựng interfaces
- Ví dụ như Buttons, Check Boxes, Radio Buttons, Text Fields, Edit Fields, Scroll Bars
- Unity cũng bao gồm các tính năng trượt ngang và trượt dọc cho việc điều khiển các giá trị cơ sở trong dữ liệu
- Các thành phần được định vị trên một mạng lưới phối hợp có liên quan đến kích thước màn hình hiện tại của các dự án đã triển khai
- Các thành phần có thể được nhóm lại để có thể khai báo hoặc sửa đổi chung

# Khai báo GUI cơ bản

```
function OnGUI () {
// Tạo một background box
  GUI.Box (Rect (10,10,100,90), "Loader Menu");

// Tạo button đầu tiên, khi người cùng click sẽ load một scene
if (GUI.Button (Rect (20,40,80,20), "Level 1")) {
Application.LoadLevel (1);
}
// Tạo button thứ 2
if (GUI.Button (Rect (20,70,80,20), "Level 2")) {
Application.LoadLevel (2);
}
}
```

# Ví dụ về GUI Box và Buttons



# Khai báo GUI Controls

- **Type (Position, Content)** – **Type** là các điều khiển, ví dụ như Button, Label,…
- **Position** đối số đầu tiên trong bất kỳ **GUI** Control nào**.**
- Đối số của chính nó được cung cấp bởi hàm **Rect()**.
- **Rect()** định nghĩa 4 thuộc tính: **left-most position**, **top-most position**, **total width**, **total height**.
- Tất cả các giá trị được cung cấp theo kiểu số nguyên (**integers)**, nơi trao đổi các giá trị pixel**.**
- Tất cả các GUI controls làm việc trong **Screen Space**, đó là độ phân giải của game được publish

# Khai báo GUI Controls

- **Content –** Là tham số thứ 2 của GUI Control, nó là nội dung thực tế được hiển thị cho điều khiển đó.
- Hầu hết trong các control, người sử dụng chỉ cho hiển thị text hoặc hình ảnh.
- Để hiển thị text, chúng ta truyền chuỗi vào đối số, ví dụ:

```
function OnGUI () {
    GUI.Label (Rect (0,0,100,50), ”Hello World");
}
```

Trong đó: **GUI.Label** chính là **Type**, **Rect (0,0,100, 50)** thể hiện **Position**
(x=0, y=0, width nằm trong khoảng 100 (pixels), height nằm trong khoảng 50 (pixels))

**Content** là “**Hello World**”

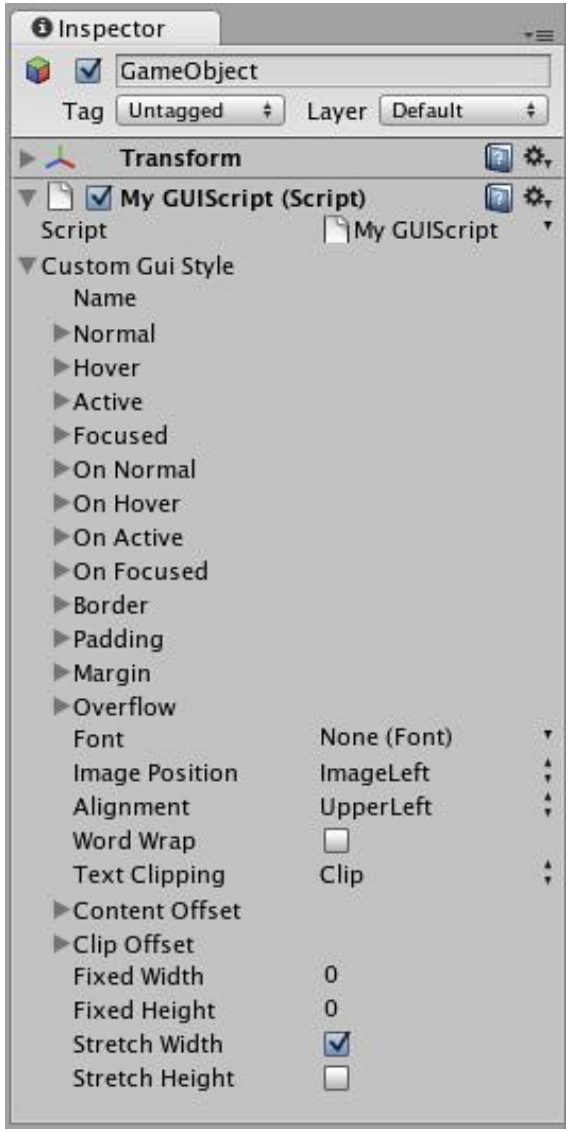
# GUI Styles và GUI Skins

- GUI Control xuất hiện đầu tiên với các **GUIStyles**.
- Mặc định khi bạn tạo một control nhưng không định nghĩa GUIStyle, unity sẽ sử dụng GUIStyle mặc định.
- GUIStyles được thiết kế bắt chước **Cascading Style Sheets** (CSS) cho trình duyệt.
- Khi định nghĩa nội dung cho các control thì một Style mặc định xuất hiện và áp dụng cho control đó. Nó cho phép bạn tạo kết hợp giống như một chức năng **Toggle** (tương tự như **Button** thông thường).
- **GUISkins** là tập hợp của **GUIStyles**. Styles định nghĩa sự xuất hiện của GUI Control. Bạn không cần sử dụng Skin nếu bạn muốn sử dụng Style.

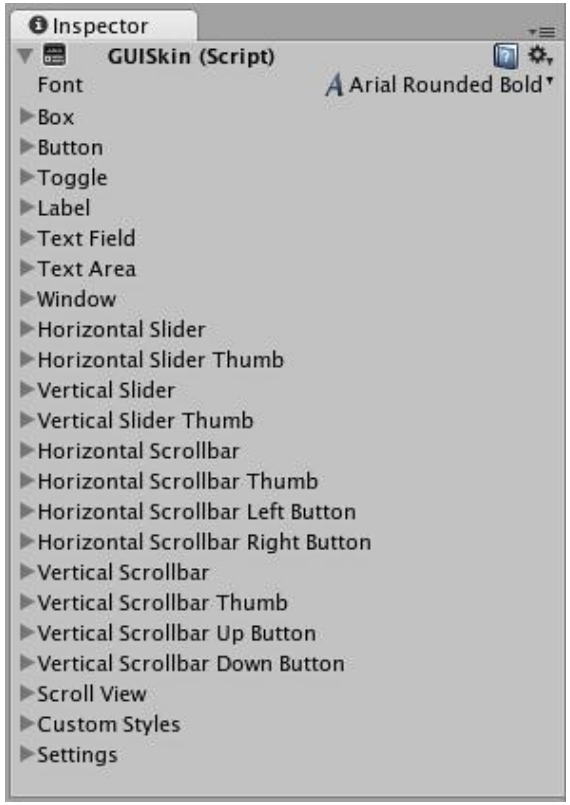
# GUI Skins

- GUI Control luôn luôn xuất hiện cùng các **GUIStyles**.
- Mặc định khi bạn tạo một mà không định nghĩa GUIStyle, Unity sẽ sử dụng GUIStyle cho sẵn.
- GUIStyles được thiết kế bắt chước **Cascading Style Sheets** (CSS) cho trình duyệt.
- Có nhiều phương pháp viết CSS khác nhau được áp dụng, bao gồm sự khác biệt về thuộc tính trạng thái theo phong cách cá nhân,cách trình bày nội dung xuất hiện.
- Control định nghĩa nội dung, Style định nghĩa cách hiển thị. Nó cho phép chúng ta tạo kết hợp giống như chức năng **Toggle** (**Button** thông thường).

# Ví dụ về GUIStyle



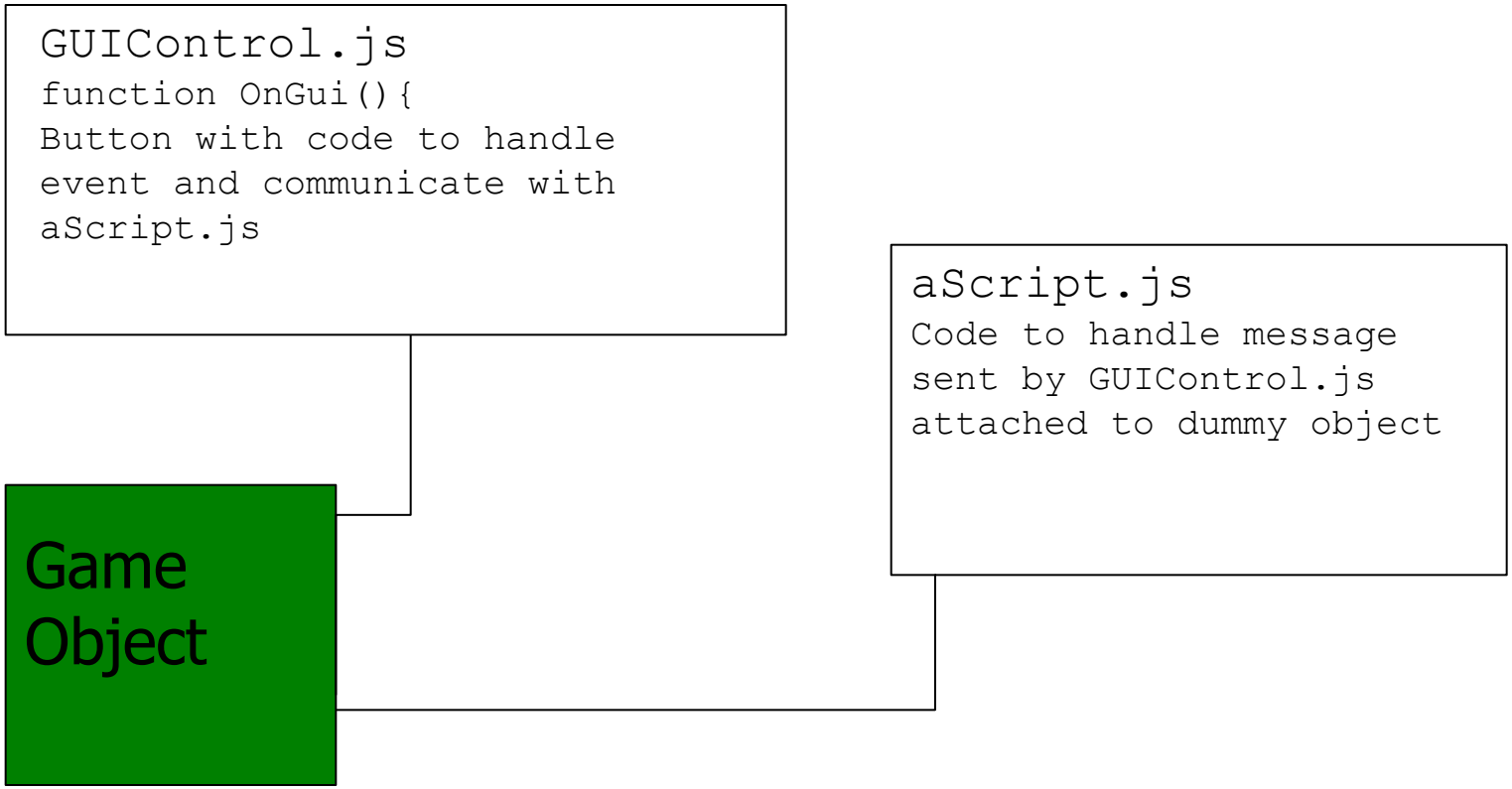
# Ví dụ về GUISkin



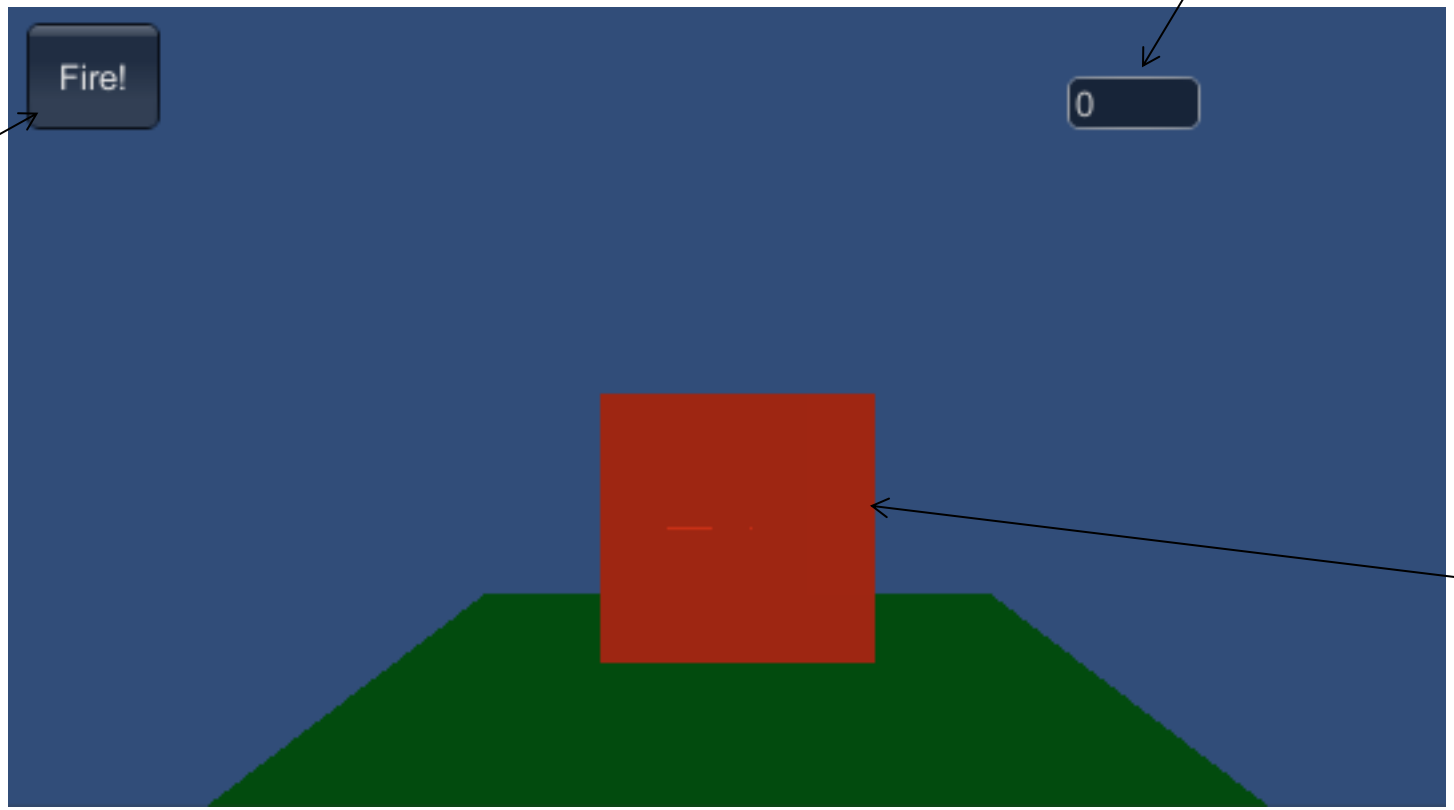
# Chức năng OnGUI ()

```
function OnGUI () {
   if (GUI.Button (Rect (25, 25, 100, 30), "Button")) {
    // code here is executed when the Button is clicked
    }
}
```

# GUI và sự tương tác với các script



# Ví dụ về GUI gọi Script

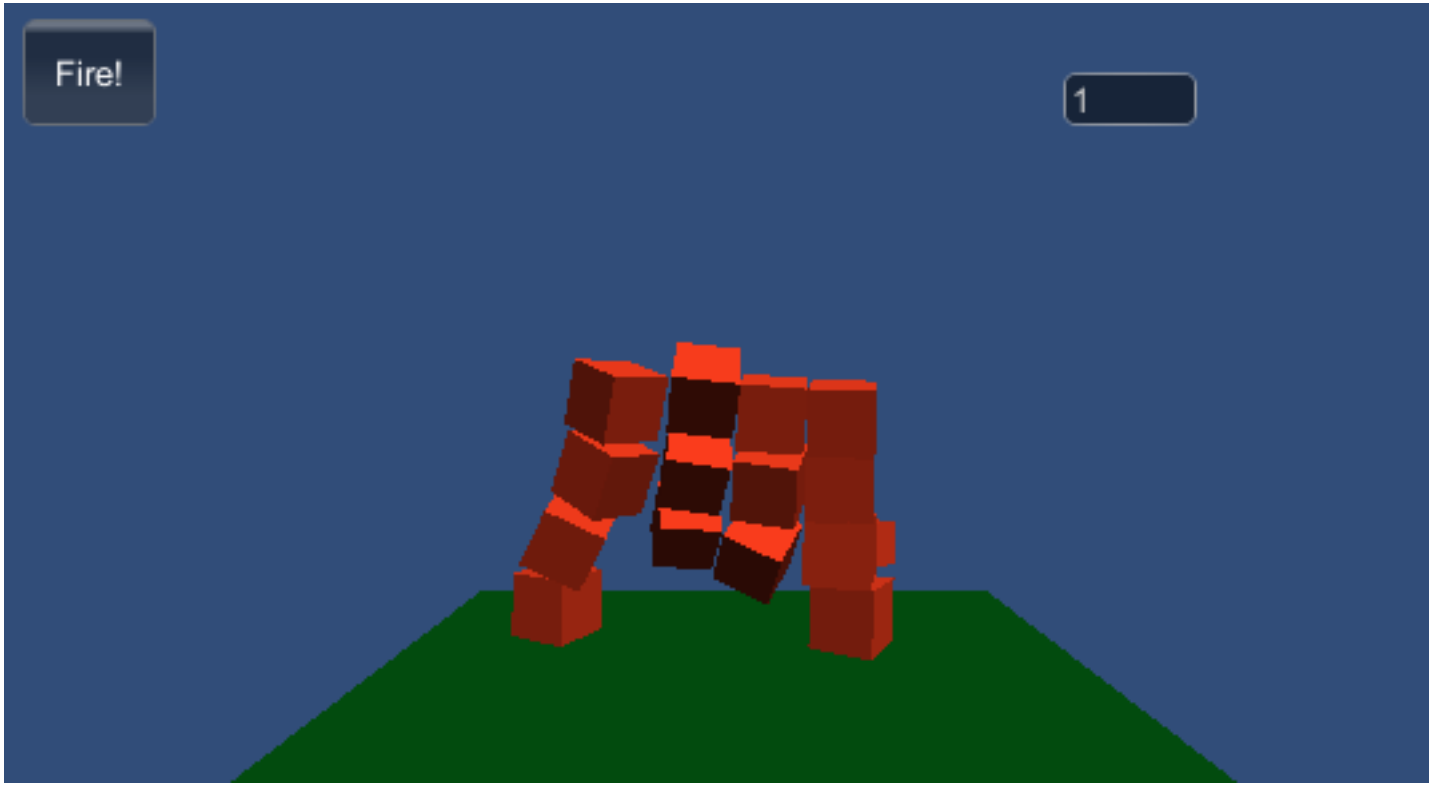



Number of Missiles
(Text Field)

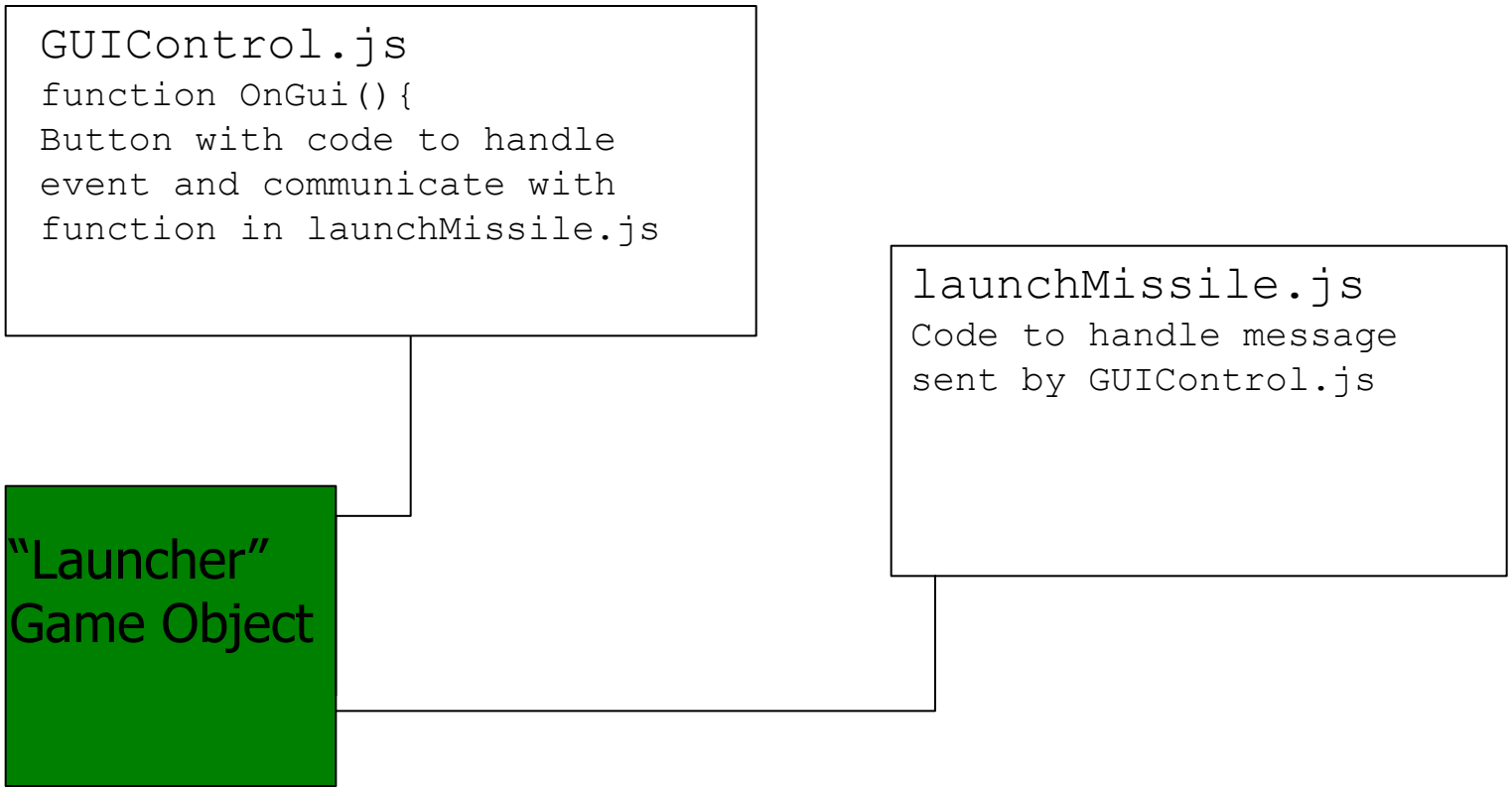
Fire!
(Button)

16 (1 x 1)
Cube
Physics Wall

Wall sưu khi click vào button Fire



# OnGui() gửi message đến hàm trong script launchMissile



# Chức năng OnGUI () trong GUIControl.js

```
// Khai báo một biến thuộc kiểu GameObject
var cannon:GameObject;

function OnGUI () {

// Gán đối tượng game với tên là ”Launcher”
cannon = GameObject.Find("Launcher");
```

# Chức năng OnGUI ()

```
if (GUI.Button (Rect (10,10,50,40), "Fire!")) {

/*
Lấy component (script) tên là launchMissile và gọi hàm chứa nó là fireMissile ()
*/
cannon.GetComponent(launchMissile).fireMissile();

}
```

Chức năng trong file launchMissile.js

```
// Hàm này được gọi khi click vào button Fire
function fireMissile(){
 //Khởi tạo đối tượng aMissile
 Instantiate(aMissile, transform.position, transform.rotation);


}
```

Cập nhật trườngTextField trong GUIControl.js

```
var numberOfMissiles:int = 0;
var missileCount:String;

GUI.TextField (Rect (400, 30, 50, 20), missileCount, 25);

function Update () {
// convert biến numberOfMissiles sang kiểu String
missileCount = numberOfMissiles.ToString();

}
```

# Tổng quan

- Graphical User Interface, viết tắt là GUI,  bao gồm tất cả các thành tố để xây dựng interfaces
- GUIStyles được thiết kế bắt chước **Cascading Style Sheets** (CSS) cho trình duyệt
- GUISkins là tập hợp của các styles.
- Mặc định, khi bạn tạo các control mà không định nghĩa GUIStyle, unity sẽ sử dụng GUIStyle có sẵn.
- Chức năng OnGUI() được sử dụng để thêm các thành phần cung cấp cho interface "look and feel".
- Kết nối với các script khác nhau từ hàm OnGUI được lưu trữ bằng cách sử dụng code tìm kiếm đối tượng game, đối tượng đó sẽ attach các script, sau đó tham chiếu đến những chức năng được yêu cầu
- `var variableName:GameObject;`
- `variableName = GameObject.Find("GameObject");`
- `variableName.GetComponent(scriptNameAttachedToGameObject).functionName();`

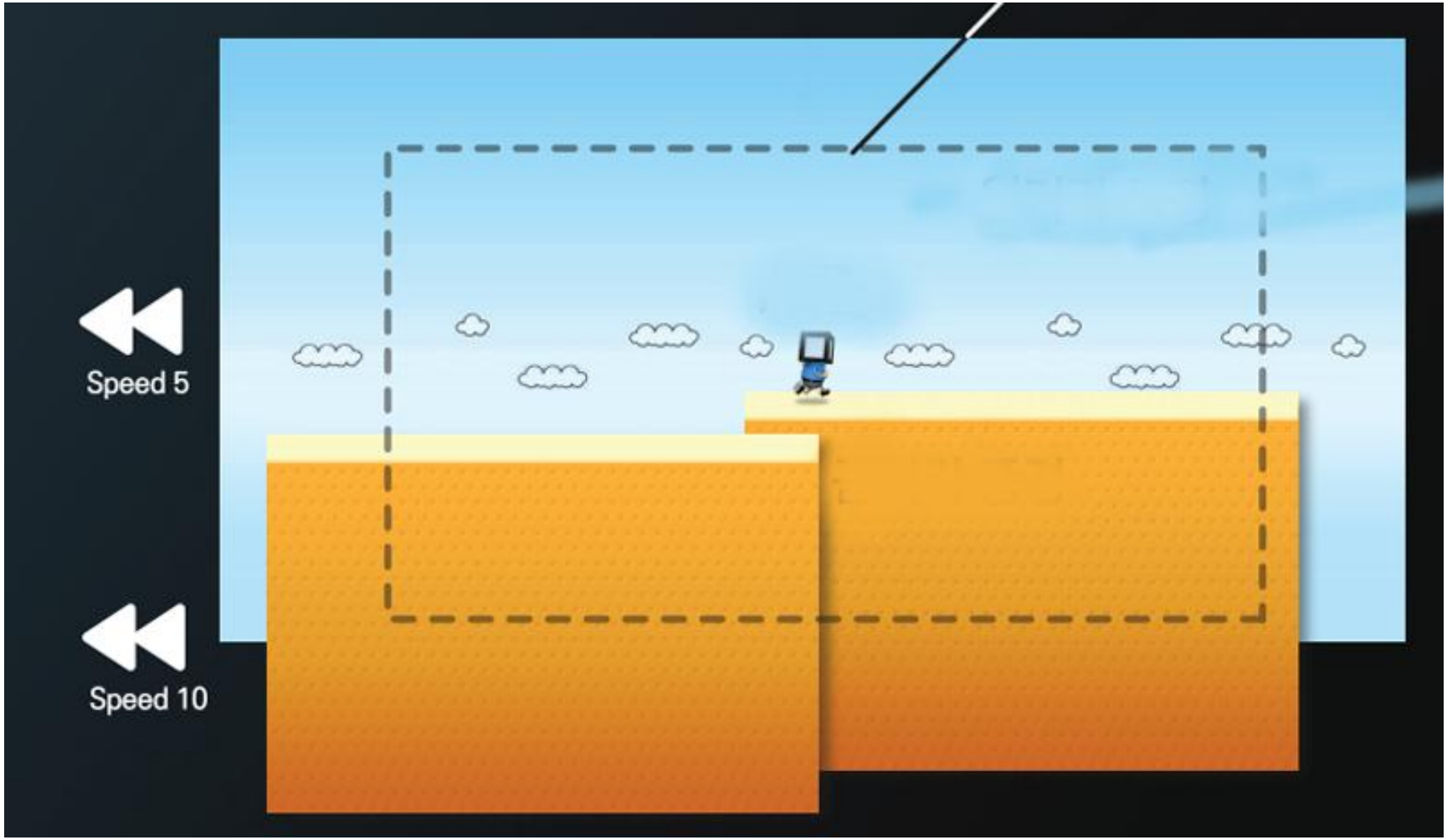
# Toolbar

- Toolbar chứa 5 loại điều khiển cơ bản, mỗi loại giữ một vai trò quan trọng trong Editor.
  - A: Transform Tool: được dung với Scene view, như quay trái , phải, lên trên, xuống dưới, phóng to thu nhỏ đối tượng.
  - B: Transform Gizmo Toggles: dung cho việc thể hiện Scene view.
  - C: Play/Pause/Step Buttons: dung cho view game, chy game ngay trong Editor để kiểm tra.
  - D: Layer Drop-down kiểm soát đối tượng nào đang được thục hiện trong Scene view
  - E: Layout Drop-down kiểm soát sự sắp xếp cảu các Views.

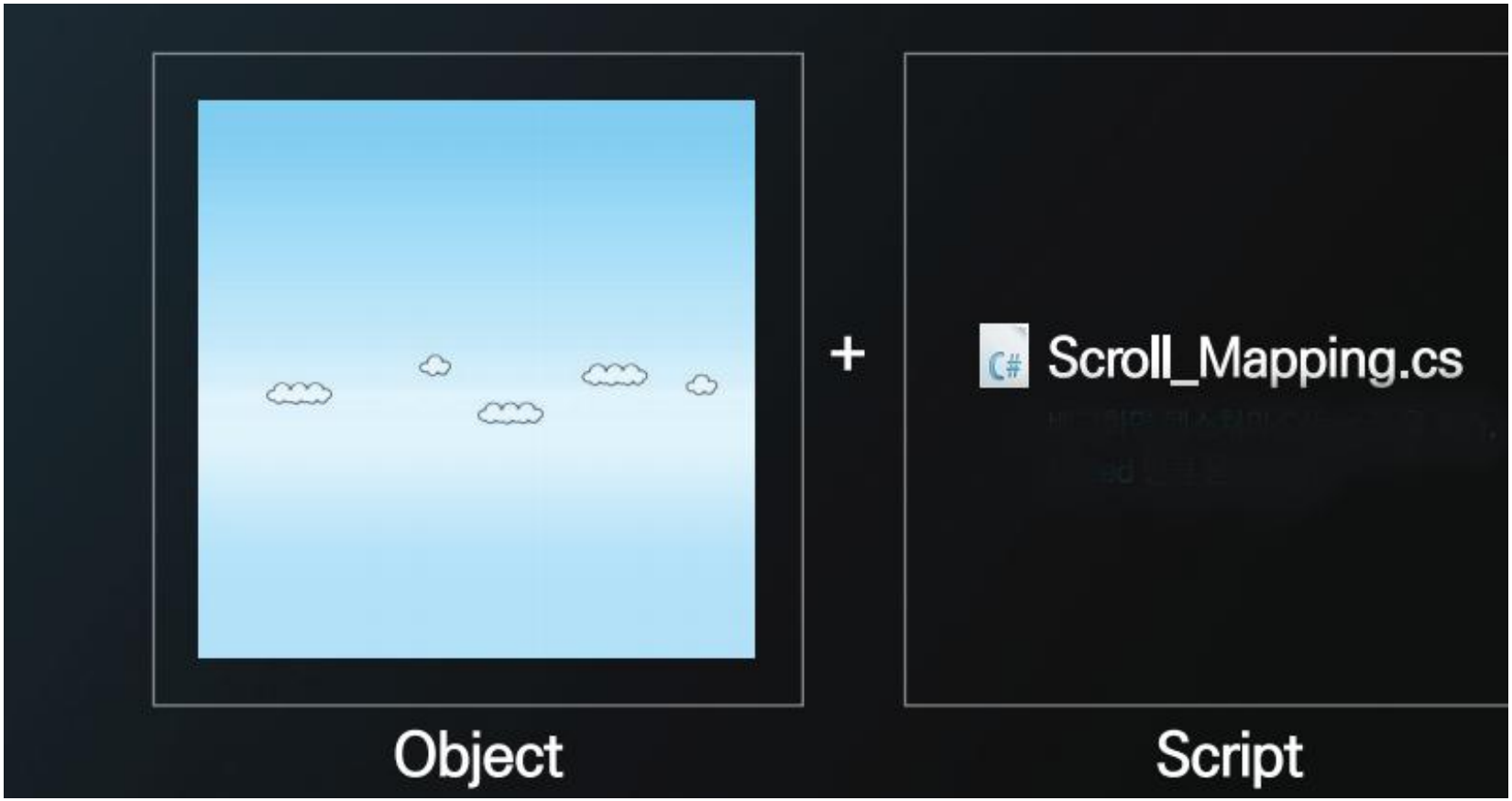

Xây dựng scene Sky:

Background chuyển động

# Mục tiêu thiết kế scene

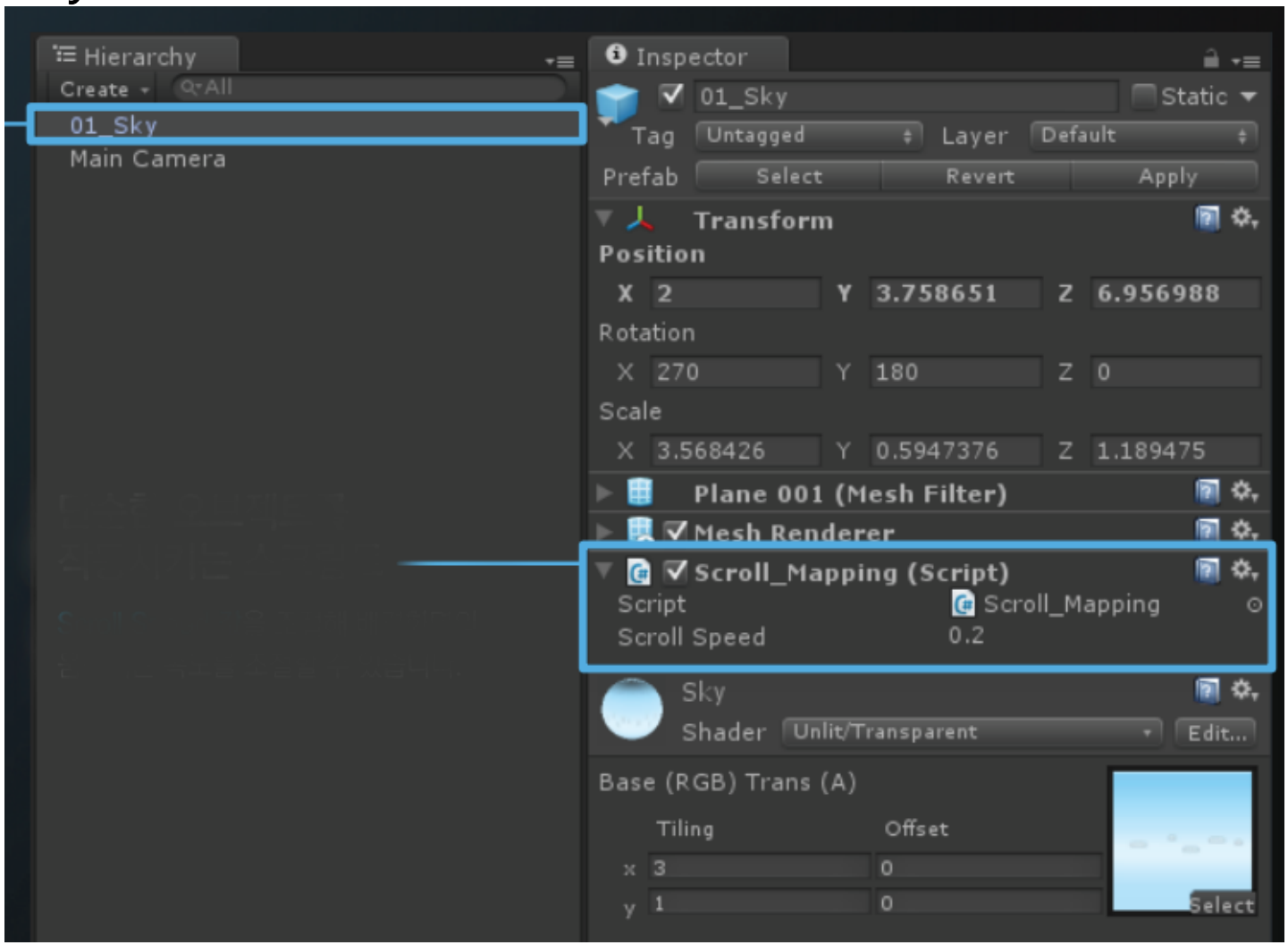


# Kết hợp đối tượng và kịch bản



# Kết hợp đối tượng và kịch bản

- Lưu ý thêm Mesh Filter từ file FBX đính kèm



# Chi tiết cà đặt Scroll_Mapping.cs

- Khai báo tốc độ của background
  - public float ScrollSpeed = 0.5f;
- Khai báo khoảng cách offset
  - float Offset;

# Chi tiết cà đặt Scroll_Mapping.cs

- Tính khoảng cách background cần di chuyển
  - Offset += Time.deltaTime * ScrollSpeed;
- Render lại background
  - renderer.material.mainTextureOffset = new Vector2 (Offset, 0.01f);

# Kết quả

- Ta được background sau chuyển động từ phải sang trái

# Kết luận

- Các thành phần của GUI
- GUI Layout
- Sử dụng Styles và Skins cho Design ‘Look and Feel’
- Script cho GUI và tương tác với đối tượng trong thế giới game

Fpt University

FPT POLYTECHNIC

THANK YOU!

www.poly.edu.vn