**Thao tác trên Unity**

* Điều khiển:
  + Con lăn để zoom in/out.
  + Đè con lăn để move around.
  + Alt + left-click để xoay xung quanh center.
* Kéo 1 hồi layout của unity bị loạn -> Góc phải trên cùng có cái pill -> Chọn Reset all layout
* Focus vào object trong tab Scene: Click vào nó bên cột Hierarchy -> nhấn F
* **Mọi thay đổi mình làm bên Inspector khi đang play game sẽ dc reset hết khi stop**

**Scene**

* : Nhấn vô để deselect all hoặc tick/untick một số thứ có trong scene ví dụ skybox,..
* : Hiển thị lưới tọa độ.
* : Chuyển qua góc nhìn của người chơi.

**Game**

* Chỉnh tỉ lệ khung hình khi zoom max cái tab Game ra nè

Graphical user interface, text, application, chat or text message

Description automatically generated

**Trục tọa độ**

* Oxyz: y là chiều cao.
* Click vào 1 trục bất kỳ (x or y or z) để nó chuyển sang góc nhìn 2D, xong nhấn vô ô vuông để nó trở thành “isometric”

A picture containing diagram

Description automatically generated

* *Nắm cái trục cho dễ*, có 3 mặt phẳng, tùy vào mình đang chọn và drag mặt nào thì điều chỉnh dc góc nhìn theo 2 trục của mặt đó:
  + Vd: Chọn Oxz thì kéo kiểu nào cũng k đổi dc chiều cao (chỉ đổi view dọc – ngang dc thôi), chọn Oxy thì dc.
* Khi drag object trên mặt phẳng nào đó, nên vào Edit -> Grid and Snap settings -> Chỉnh x, y, z theo ý mình để khi Ctrl + drag nó lọt đúng vào bội của x, y, z.

Graphical user interface, application

Description automatically generated

**Camera (object Main Camera)**

* Thay vì background bầu trời thì Clear Flags -> Solid Color rồi chọn màu.

**Phím tắt**

* Graphical user interface

  Description automatically generated Từ trái qua W E R T, hay xài Move (W) và Rotation (E) nhất.\
* Ctr + D để duplicate object
* Asset store: ctrl + 9
* Phóng to cửa sổ: Shift + Space (nhớ click/hover vào cửa sổ đó trước)
* Play: Ctrl + P.

**Inspector - Các component của 1 object**

* Transform (Có thể đè vào X, Y, Z rồi nắm kéo để dễ căn chỉnh hơn là nhập số)
  + Vị trí
  + Góc xoay
  + Kích thước
* Mesh Renderer: Hiện object ra màn hình.
* Box Collider: Sự va chạm giữa các physic object, nếu bỏ tick thì object đó nó sẽ đi xuyên qua các object khác.
* Mesh Filter: Chọn shape cho object (Hình khối, hình tròn, hình trụ..)

**Material**

* Smoothness càng cao thì object nó càng bóng.

**Prefab (biến cái object mình xài nhiều thành 1 kiểu asset để tái sử dụng)**

* Kéo object xuống khung assets.
* Nếu chỉnh bất cứ stat gì trên prefab nằm trong asset này, thì tất cả thay đổi sẽ dc apply lên các object khác có cùng prefab.
* Muốn chỉnh stat riêng cho từng clone thì chỉnh bên Inspector của nó như bth, còn muốn apply lên nhiều anh em của nó thì đè Ctrl để chọn các anh em r chỉnh.

**Layers**

A screenshot of a computer

Description automatically generated with low confidence

* Mình block layer nào thì trong scene k thể click vào các object thuộc layer đó dc nữa (click bên Hierarchy thì dc).

**Tạo sương mù**

* Để k cho người chơi nhìn thấy các obstacle ở xa.
* Window -> Rendering -> Lighting -> Kéo tab này qua cột của Inspector để nhìn cho dễ -> Tab Environment -> Tick ô Fog -> Chọn màu cho fog (Tốt nhất là dùng color picker để nó trùng với màu của môi trường xung quanh).

**Xử lý lỗi va chạm**

* Đôi khi object player nó đi xuyên qua các object obstacle luôn.
* Rigidbody -> Collision Detection -> Chuyển từ Discrete sang Continous.
* Làm tương tự với lũ clone của prefab obstacle.
* Edit -> Project Settings -> Time -> Giảm Fixed Timestep từ 0.02 về 0.01 để Unity nó check collision thường xuyên hơn -> Điều này sẽ ảnh hưởng tới các force movement nên nhớ để ý rồi tăng lực lên.

**Text (Object Text)**

* Hierarchy -> + -> UI -> Text.
* Horizontal overflow -> Đổi Wrap sang Overflow để nó có tràn khỏi khung text thì nó tràn ngang chứ k xuống hàng.
* Canvas -> UI Scale Mode -> Chọn Scale with screensize để cái text không bị cố định pixel.
* **Chữ bị vỡ nét, phải làm sao? Check coi còn cái gì ngoài text vỡ nét k, nếu mọi thứ đều vỡ thì chỉnh Scale bên tab Game về 1x.**

**Lỗi**

* The reference assemblies for .NETFramework,Version=v4.7.1 were not found
* Cài .NET SDK -> Cài đúng cái .NET devpack 4.7.1 cho nó, **PHẢI LÀ 4.7.1**. Nếu chưa dc thì cài thêm mono.
* Nó hiện lỗi liên quan tới quyền or connection này nọ thì cứ Run as admin..

**Font chữ**

* Tạo folder Fonts trong Assets.
* Import các font chữ tải từ google về vào đây.
* Sau đó trong phần Font của object Text cứ chọn font thoải mái.

<https://www.stdio.vn/unity-game/cac-thanh-phan-co-ban-trong-unity-Cf3Uw1>

Tạo Scene dưới mục Asset để có để đặt tên cho scene.

Link học <https://www.youtube.com/watch?v=IlKaB1etrik&list=PLPV2KyIb3jR5QFsefuO2RlAgWEz6EvVi6&index=2>

**Coding**

**Unity**

* Click vào object -> Nhìn qua Inspector -> Add component -> New script -> Double click vào script để nó vở visual studio lên -> Code -> Ctr + S -> Quay lại Unity cho nó đồng bộ hóa cái code vừa save -> Nhấn play bên Unity.
* Sau khi khai báo biến -> Vào Unity -> Bên Inspector kéo component (Hoặc kéo object bên Hierarchy vào cũng dc luôn) vừa khai báo trong code vào chỗ tên biến trong phần Script (Biến phải public)
* Syntax suggestion trong visual studio
* A screenshot of a computer

  Description automatically generated with medium confidence

**Chung**

* Biến kiểu float khi gán phải thêm chữ f vào (kẻo Unity complains). Vd: float forwardForce = 2000f.
* Debug.Log(“Hellooooo”): in ra console của Unity.
* Input.GetKey(“d”): trả về true/false.
* Code cái gì liên quan tới UI vd Text thì phải khai báo

using UnityEngine.UI;

* Tìm component trong object

GetComponent<PlayerMovement>().enabled = false;

* Tìm object thuộc loại

FindObjectOfType<GameManager>().EndGame();

**Hàm**

* Start(): Được gọi đúng 1 lần khi start game (Start is called before the first frame update)
* Update(): Mỗi frame hình gọi 1 lần (Update is called once per frame), **tùy vào tốc độ render frame của PC mà tốc độ gọi hàm này cũng thay đổi theo, không cố định**.
* Dùng Time.deltaTime (the amount of seconds since the computer dew the last frame) = 1/số frame được render trong 1s
  + Vd: rg.AddForce(0, 0, 2000 \* Time.deltaTime);
    - PC render được 10 frames/s -> Time.deltaTime = 0.1 -> Mỗi 0.1s add thêm 200 force -> Sau 1s add tổng cộng 2000 force.
    - PC render được 20 frames/s -> Time.deltaTime = 0.05 -> Mỗi 0.05s add thêm 100 force -> Sau 1s add tổng cộng 2000 force.
* FixedUpdate(): Mọi code liên quan đến physics nên nằm trong hàm này (Unity likes this a lot)

**Rigidbody**

* rg.useGravity = false/true;
* rg.AddForce(value on x, value on y, value on z, {ForceMode}).
  + Mặc định ForceMode.Force: add force dựa trên mass của object.
  + Mọi object di chuyển đều tạo ra momentum, nên để thay đổi quỹ đạo di chuyển nếu dùng ForceMode.Force thì phải chờ nó add đủ force để đổi quỹ đạo.
  + Dùng ForceMode.VelocityChange thì nó ngay lập tức thay đổi vận tốc, bỏ qua mass luôn.
* Trong Unity, Rigidbody -> Constraints -> Có thể tick vào các ô tương ứng để khóa chuyển động (tick rồi chạy thử là biết).
* Khi tác động lực, object bị nảy lên rồi lệch hướng dần là do ma sát -> Vd muốn nó lăn thẳng đều hoài thì chọn 1 trong 2 cách:
  + Set constraint trong Rigidbody
  + Thêm Physics Material vào cho object Ground rồi sửa Dynamic friction, Static friction về 0.
* Drag: Lực cản không khí
  + Vd: khi player tông vào tường, nếu k có drag thì player sẽ bị trôi đi 1 đoạn rồi mới dừng hẳn -> nhìn k real, nếu có drag vào thì nó sẽ dừng lại nhanh hơn (do lực cản khiến v của player về 0 lẹ hơn).

**Camera**

* Muốn camera nó lăn lộn, nghiêng ngả theo movement của object thì kéo cho Main Camera thành con của object đó.
* Muốn chỉnh góc nhìn từ sau lưng object, trong hàm Update():
  + transform.position = player.position + offset;
    - player có kiểu Transform
    - offset có kiểu Vector3 (dùng để camera nó k bị lọt vào bên trong object -> biến thành góc nhìn thứ nhất).

**Kiểm tra va chạm**

* Add script cho object

public class PlayerCollision : MonoBehaviour

{

public PlayerMovement playerMovement; //PlayerMovement here is script PlayerMoverment.cs

private void OnCollisionEnter(Collision collision)

{

if (collision.collider.tag == "Obstacle")

{

Debug.Log("We hit obstacle");

playerMovement.enabled = false; //Make player stop moving when collision happens

}

}

}

* Add tag cho object chướng ngại vật (dùng collider.tag sẽ nhanh hơn là dùng collider.name vì system nó phải search qua tất cả object).

**Game Manager**

* Là 1 empty object đính kèm script để điều khiển các công việc như hiện Start scene, chuyển Level scene,..
* Muốn chuyển scene or reload scene phải khai báo

using UnityEngine.SceneManagement;

1. **EndGame()**

* Giả sử mình có cái GameManager.cs như này nhé

public class GameManager : MonoBehaviour

{

bool gameHasEnded = false;

public float restartDelayTime = 1f; //1 second

public void EndGame()

{

if (gameHasEnded)

return;

gameHasEnded = true;

Debug.Log("GAME OVER");

Invoke("Restart", restartDelayTime);

}

public void Restart()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

}

}

* Giờ muốn nó log ra console “GAME OVER” khi end game có 2 cách (Cách 1 performance tốt hơn)

1. Vào PlayerCollision.cs

public class PlayerCollision : MonoBehaviour

{

public PlayerMovement playerMovement;

public GameManager gameManager;

private void OnCollisionEnter(Collision collision)

{

if (collision.collider.tag == "Obstacle")

{

Debug.Log("We hit obstacle");

playerMovement.enabled = false; //Make player stop moving when collision happens

gameManager.EndGame();

}

}

}

* Nhưng lỡ sau khi game over, mình muốn spawn player lại để chơi tiếp thì sao? Khi này object Player bị xóa mất tiêu rồi -> Mất reference của biến gameManager – object GameManager -> Rồi giờ lấy đâu GameManager để hiện “GAME OVER” nữa?

1. Tạo prefab cho object Player

public class PlayerCollision : MonoBehaviour

{

public PlayerMovement playerMovement;

private void OnCollisionEnter(Collision collision)

{

if (collision.collider.tag == "Obstacle")

{

Debug.Log("We hit obstacle");

playerMovement.enabled = false; //Make player stop moving when collision happens

FindObjectOfType<GameManager>().EndGame();

}

}

}

1. **Restart()**

public void Restart()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

}

* Muốn chờ một tí rồi mới gọi hàm Restart() để xem ending movement thì xài

Invoke("Restart", restartDelayTime);

* Sau khi restart (reload scene), thấy màu của các object lạ lạ là do lighting bị bủh.
  + Window -> … -> Lighting -> Tab Scene -> Ở dưới cùng nhấn Generate Lighting (Muốn thấy sự bủh trở lại thì xổ mũi tên ra -> Chọn Clear baked data) và bỏ tick ô Auto..
  + Điều này là do
    - Nếu tick auto thì là do khi mình reload scene lẹ quá Unity nó chưa kịp điều chỉnh ánh sánh cho mình (Brackeys nói vậy, [link](https://youtu.be/VbZ9_C4-Qbo?list=PLPV2KyIb3jR5QFsefuO2RlAgWEz6EvVi6&t=559))
    - Còn Unity bản mới nó k có tick, nhưng chắc lý do cũng giống trên, nên phải xài cái này để nó gen lighting ra trước rồi lưu vào cache

Graphical user interface, text, application, chat or text message

Description automatically generated

* Lưu ý add scene vào Build Settings, [link](https://youtu.be/VbZ9_C4-Qbo?list=PLPV2KyIb3jR5QFsefuO2RlAgWEz6EvVi6&t=659)

Background pattern

Description automatically generated

* + Nhớ add hết scenes vào trước khi Export.
  + Tùy theo thứ tự tick các scene mà Unity nó sẽ quyết định build scene nào trước.