TRƯỜNG ĐẠI HỌC CẦN THƠ

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

KHOA KHOA HỌC MÁY TÍNH

NIÊN LUẬN CƠ SỞ NGÀNH KHOA HỌC MÁY TÍNH

ĐỀ TÀI:

XÂY DỰNG GAME 2D TRÊN UNITY

Giảng viên hướng dẫn:
Ths. Huỳnh Ngọc Thái Anh

Sinh Viên Thực Hiện:
1. Nguyễn Quốc Trầm B1913276
2.Nguyễn Xuân Trúc B1913277

Cần Thơ, tháng 11/2022

LỜI CÁM ƠN

Để có được bài niên luận này, chúng em xin được bày tỏ lòng biết ơn chân thành và sâu sắc đến thầy Huỳnh Ngọc Thái Anh – người đã trực tiếp cung cấp những kiến thức và hướng dẫn chúng em. Trong suốt quá trình thực hiện niên luận, nhờ những sự chỉ bảo và hướng dẫn quý giá đó mà chúng em có thể hoàn thành đề tài này.

Chúng em cũng xin gửi lời cám ơn chân thành đến các Thầy Cô Giảng viên Đại học Cần Thơ, đặc biệt là các Thầy Cô ở Trường CNTT & TT, những người đã truyền đạt những kiến thức quý báu trong thời gian qua.

Chúng em cũng xin chân thành cảm ơn bạn bè cùng với gia đình đã luôn động viên, khích lệ và tạo điều kiện giúp đỡ trong suốt quá trình thực hiện để chúng em có thể hoàn thành bài đề tài một cách tốt nhất.

Tuy có nhiều cố gắng trong quá trình thực hiện đề tài, nhưng không thể tránh khỏi những sai sót. Chúng em rất mong nhận được sự đóng góp ý kiến quý báu của Thầy để bài niên luận hoàn thiện hơn.

Cần Thơ, ngày 1 tháng 12 năm 2020

Người viết

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

Trang

DANH MỤC HÌNH

CHƯƠNG I: TỔNG QUAN

1.1 Lý do chọn đề tài

Trong thời đại công nghệ thông tin như hiện nay, sản phẩm công nghệ ngày càng chịu sự đánh giá khắt khe hơn từ phía những người dùng, đặc biệt là về sản phẩm Game được nhận rất nhiều sự đánh giá từ phía các Game thủ, hay chỉ là những người chơi bình thường.

Ngành công nghiệp Game hiện nay có thể nói là bùng nổ, với tốc độ phát triển đến chóng mặt, rất nhiều những Game hay và hấp dẫn đã được ra đời trong thời gian qua. Phía sau những Game phát triển và nổi tiếng như vậy đều có một Game Engine. Game Engine là một công cụ hỗ trợ, một Middleware giúp người phát triển viết Game một cách nhanh chóng và đơn giản, đồng thời cung cấp khả năng tái sử dụng các tài nguyên và mã nguồn cao.

Từ xu hướng phát triển và những bất cập trên, đồ án này sẽ khảo sát và nghiên cứu về Engine Unity – một Game Engine rất phổ biến và không kém mạnh mẽ hiện nay nhằm thực nghiệm việc phát triển game trên nhiều nền tảng. Chuẩn bị kiến thức và kỹ năng cho định hướng nghề nghiệp (phát triển Game) sau này của chúng em, góp phần vào sự phát triển của ngành công nghiệp Game nước nhà. Các chương tiếp theo sẽ giới thiệu về Engine Unity bao gồm các đặc điểm, tính năng, công cụ và thành phần trong Engine này. Sau những nội dung về thiết kế, tài liệu sẽ trình bày về việc lập trình trên Unity khi giới thiệu về ngôn ngữ lập trình cùng các lớp, hàm trong thư viện dựng sẵn của Unity thông qua các ví dụ thực tế khi phát triển một game bắn máy bay 2D.

Cuối cùng là giới thiệu về bối cảnh, cốt truyện, tài liệu hướng dẫn sử dụng của trò chơi Demo cùng chương tổng kết về các khó khăn và những thành quả trong suốt quá trình phát triển trò chơi Demo trên Unity.

1.2 Phạm vi và mục tiêu đề tài

Tìm hiểu, nghiên cứu cơ chế, cách thức hoạt động, thực hiện cài đặt và sử dụngphần mềm làm game Unity. Kết hợp cùng với ngôn ngữ lập trình C# để thực hiện đề tài này.

Mục tiêu là xây dựng thành công game 2D trên nền tảng Unity

1.3 Phương pháp tiếp cận

- Đọc các tài liệu, hướng dẫn về lập trình game Unity trên các nền tảng internet.
- Tìm hiểu và cài đặt phần mềm Unity, xây dựng trò chơi.
- Rút ra kinh nghiệm thực tế từ quá trình thực hiện đề tài.

1.4 Kế hoạch thực hiện

TUẦN	Công việc
1-2	Tìm hiệu lập trinnh Unity và nhận đề tài
2-3	Nhận đề tài và bắt đầu thực hiện
3-4	Nghiên cứu tổng quan về Unity Engine
4-5	Hoàn thành thiết kế cơ bản của tựa game 2D
5-6	Xây dựng và tìm hiệu về chức năng của các cửa sổ trong Unity
6-7	Hoàn thành các bẫy và cách duy chuyển của nhân vật.
7-8	Thêm các chức năng cơ bản của game.
8-9	Sửa lỗi do các chức năng được thêm vào.
9-10	Tìm hiểu và xây dựng các màn chơi, sửa lỗi nhỏ.
10-11	Hoàn thiện chức năng Game Play của game.
11-12	Xây dựng GameManager cho game.
12-13	Sửa lỗi của GameManager.
13-14	Thêm chức Restart cho game. Tìm hiểu vầ tiếp tục xậy dựng màn chơi.
14-15	Hoàn thành Game và bài niên luận. Sửa lỗi.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về Unity

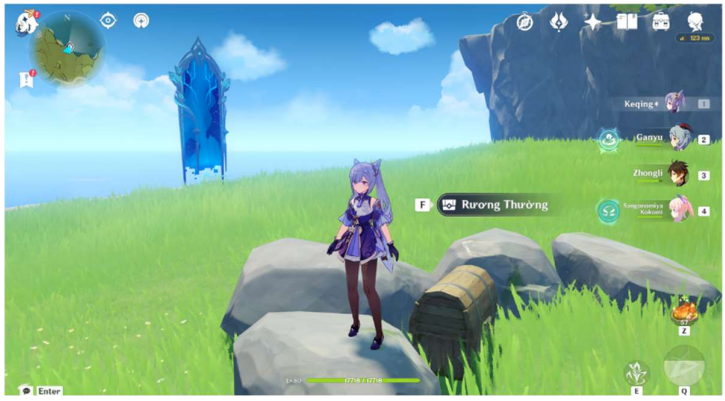
2.1.1 Unity là gì?

Unity là một phần mềm làm game đa nền tảng, các nền tảng được hỗ trợ hiện nay là Android, IOS, Linux, macOS, Windows,,..

Unity3D cung cấp một hệ thống toàn diện cho các lập trình viên, từ soạn thảo mã nguồn, xây dựng công cụ tự động hóa đến trình sửa lỗi nên cũng khá dễ sử dụng. Ngôn ngữ lập trình chính của Unity là C#, ngoài ra còn có hỗ trợ cho Javascript.

Unity cũng tận dụng chức năng của các thư viện phần mềm như engine mô phỏng vật lý PhysicX của Nvidia, OpenGL và Direct3D để kết xuất hình ảnh 3D, OpenAL cho âm thanh, … nên nó hỗ trợ rất mạnh cho công việc lập trình game.

Các game được lập trình bằng Unity như Genshin Impact (mihoyo), Pokemon GO (Niantic),..



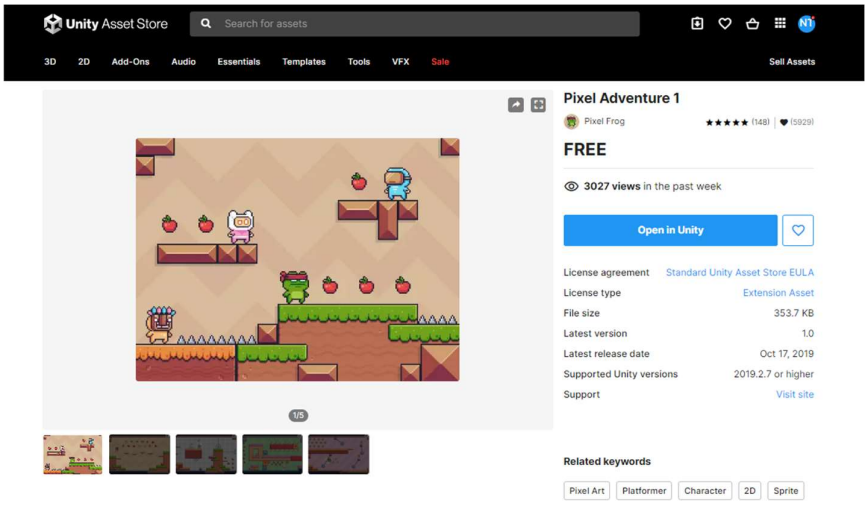
Hình 1 Giao diện trò chơi Genshin Impact

2.2 Tổng quan về các thành phần trong Unity

2.2.1 Assets

Assets là những tài nguyên được sử dụng trong việc phát triển game trong Unity.

Các Assets này có thể là hình ảnh, mô hình 3D, âm thanh, hiệu ứng, ... được tạo ra bởi các nhà phát triển, có thể được download miễn phí hoặc trả phí. Tính năng này giúp giảm thiểu rất nhiều thời gian cho việc thiết kế và lập trình game.

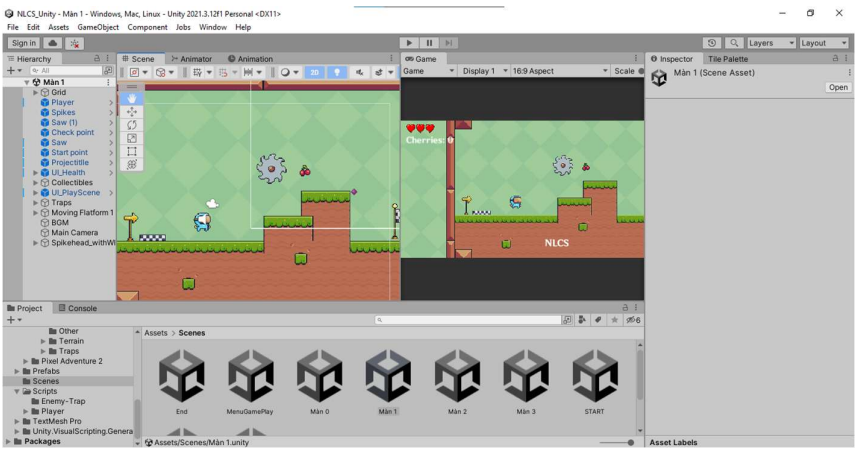


Hình 2 Cửa hàng Unity Asset Store

2.2.2 Scenes

Trong Unity, Scene là một cảnh game, không gian game nơi thiết lập bố cục của các GameObjects, hoặc là một phần chứa các thiết lập giao diện như các menu trong game.

Tạo ra nhiều Scenes sẽ giúp linh hoạt trong việc phân phối tối ưu tài nguyên, quản lý các phân đoạn trong game một cách độc lập.



Hình 3 Một scene trong trò chơi

2.2.3 Game Object

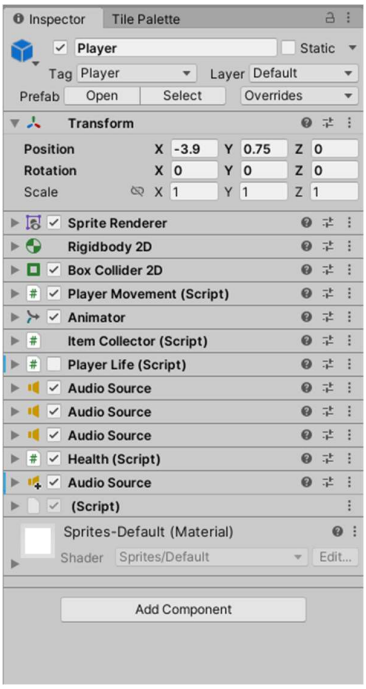
Unity định nghĩa GameObject là đối tượng đại diện cho các Assets trong game như nhân vật, cây cỏ, công cụ, đạo cụ, camera, hiệu ứng ... Các GameObject đều chứa một thuộc tính cơ bản là Transform, dùng để thiết lập vị trí, hướng và kích thước của GameObject.

Các object được xây dựng và sử dụng trong đề tài:

STT	Tên Object	Mô tả	Hình ảnh
1	Player	Nhân vật người chơi điều khiển trong game.	
2	Saw	Bẫy trong game.	
3	Spikes	Bẫy trong game.	
4	Spike Head	Bẫy trong game.	
5	Falling Platforms	Nền tảng sẽ rơi khi nhân vật chạm vào sau khoảng thời gian.	
6	Platforms	Nền tảng giúp nhân vật di chuyển qua địa hình.	
7	Background	Hình nền trong game.	
8	Terrain	Địa hình trong game.	
9	Checkpoints	Vị trí băt đầu/ kết thúc.	
10	Cherries	Vật phẩn trong game	
11	Jump Pad	Làm cho nhân vật nhảy cao hơn	
12	Tim	Vật phẩn trong game	

2.2.4 Components

Components là các thuộc tính thêm vào GameObject như là Animation, âm thanh, mô hình 3D, hiệu ứng, … nhằm xây dựng, kết hợp các yếu tố khác nhau để xác định các hình thái, hành vi, hiệu ứng, ... của đối tượng mong muốn trong game.



Hình 4 Cửa sổ Inspector

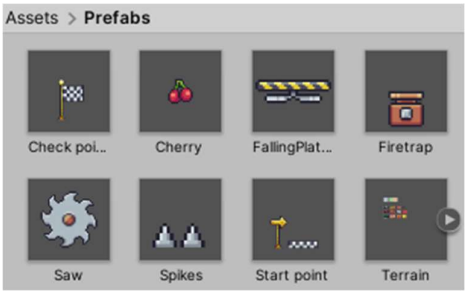
2.2.5 Scripts

Scripts là một Component trong Unity. Đây là thành phần thiết yếu dùng để tương tác với các hành động 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 ngôn ngữ C#.

2.2.6 Prefabs

Prefabs là một GameObject hoàn chỉnh sau khi thêm các thuộc tính và được lưu trữ lại để tái sử dụng. Các GameObject được nhân bản từ một prefab sẽ giống nhau hoàn toàn.

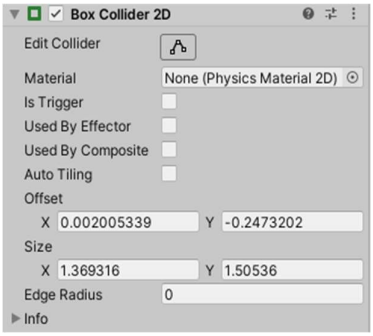


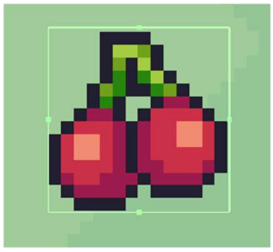
Hình 5 Prefabs trong trò chơi

2.2.7 Collider

Collider là vật mà hệ thống vật lý dùng để nhận biết va chạm mỗi khi hai đối tượng bất khi va vào nhau. Các collider có các hình dạng đơn giản như: BoxCollider, SphereCollider, CapsuleCollider, MeshCollider, PhysicMaterial, Rigidbody. Trong đề tài các loại được sử dụng là:

Box collider: Sử dụng cho các đối tượng có dạng hình chữ nhật.
Circle collider: Sử dụng cho các đối tượng có dạng hình tròn.
Circle collider: Sử dụng cho các đối tượng có địa hình.



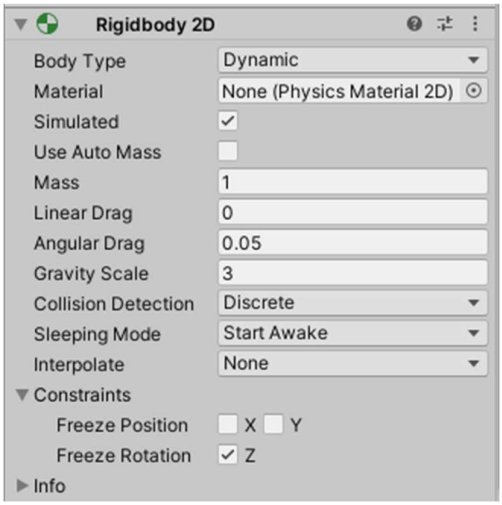


Hình 6 Cửa sổ Box Collider 2D và ví dụ

- Các thông số:
+ Material: tham chiếu đến physic material.
+ Is Trigger: dạng true/false, cho phép va chạm có đi xuyên qua không?
+ Used by Effector: nếu check thì va chạm vẫn hoạt động dù có đính kèm effector.
+ Used by Composite: Collider có được điều khiển bởi Composite hay không.
+ Auto Tiling: Xác định xem hình dạng của BoxCollider2D có được cập nhật tự động hay không dựa trên các thuộc tính của SpriteRenderer.
+ Offset: Vị trí va chạm
+ Size: điều chỉnh kích thước theo các trục tương ứng.

2.2.6 Rigidbody

Hệ thống mô phỏng vật lí trong game.



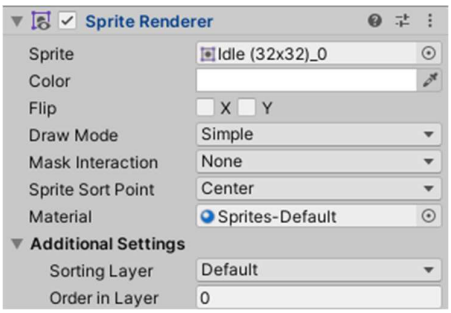
Hình 7 Cửa sổ Rigidbody

- Mass: khối lượng (đơn vị tùy ý).
- Drag: Sức cản không khí ảnh hưởng như thế nào với đối tượng khi di chuyển. 0 có nghĩa là không có sức cản không khí, và vô cùng làm cho các đối tượng di chuyển ngay lập tức dừng lại.
- Angular Drag: Sức cản không khí ảnh hưởng đến các đối tượng khi quay từ mô-men xoắn. 0 có nghĩa là không có sức cản không khí. Không thể làm cho vật dừng quay hẳn chỉ bằng cách thiết lập Angular Drag của nó đến vô cùng.
- Use Gravity: Nếu được kích hoạt, các đối tượng bị ảnh hưởng bởi lực hấp dẫn.
- Is Kinematic: Nếu được kích hoạt, các đối tượng sẽ không được thúc đẩy bởi động cơ vật lý.
- Interpolate: giảm xóc.
- Collision detection: Được sử dụng để ngăn chặn các đối tượng chuyển động nhanh qua các đối tượng khác mà không phát hiện va chạm.
- Constraints: Ràng buộc Những hạn chế về chuyển động của Rigidbody:

2.2.7 Sprite

- Là một hình ảnh 2D của một game object có thể là hình ảnh đầy đủ, hoặc có thể là một bộ phận nào đó. Unity cho phép tùy chỉnh màu sắc, kích thước, độ phân giải của một hình ảnh 2D.

- Sprite Renderer: là một composite hỗ trợ xuất ra hình ảnh 2D cho 1 object rỗng.



Hình 8 Cửa sổ Sprite Renderer

+ Sprite: Hình ảnh 2D cần xuất ra.

+ Color: Màu sắc.

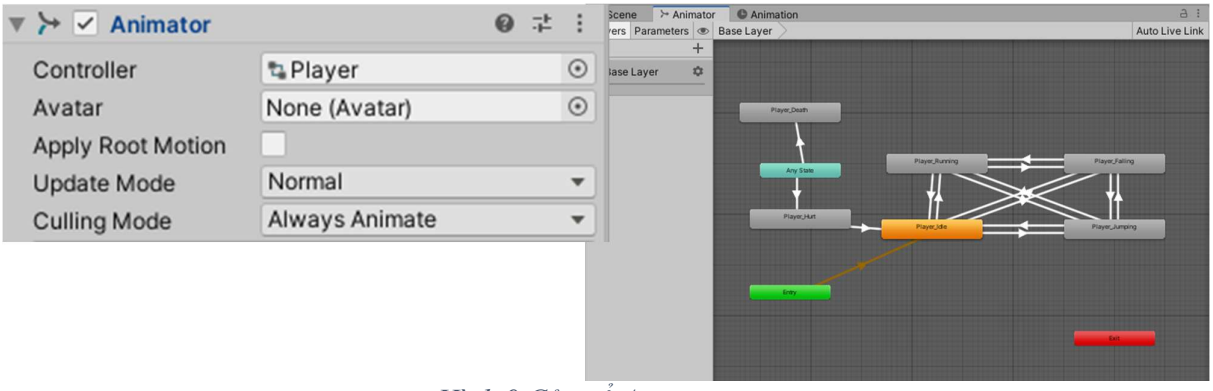
+ Flip: Lật sprite trên trục X/ Y.

+ Draw Mode: Chế độ vẽ.

+ Mask Interaction: Cách sprite tương tác với mặt nạ.

2.2.8 Animator

- Trong 2D thi animotion là tập một hình ảnh động dựa trên sự thay đổi liên tục của nhiều sprite khác nhau. Trong 3d là một tập hợp các sự thay đổi theo thời gian của đối tượng trong không gian. Mỗi thay đổi là một key frame.-Key Frame hay Frame là một trạng thái của một animation.



Hình 9 Cửa sổ Animator

+ Controller: Bộ điều khiển animation gắn liền với nhân vật này. Nó sẽ quản lý các animation clip, các thông số tốc độ của animation, thứ tự các clip…

+ Avatar: thành phần tạo hình ảnh cho object.

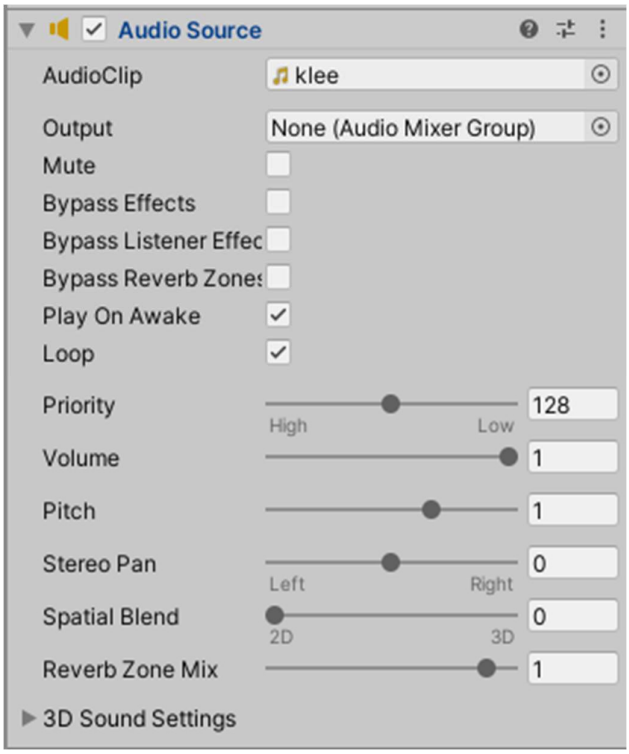
+ Apply Root Motion: dạng true-false, cho phép thiết lập animation có di chuyển theo không gian đã được tạo khi cấu hình animation.

+ Animate Physics: dạng true-false, khi được chọn, các hình ảnh trong animation sẽ có thể tương tác vật lý với nhau.

+ Culling mode: chọn chế độ cho hình ảnh động.

2.2.9 Audio Source

Audio Source là 1 built-in component được Unity xây dựng sẵn để hỗ trợ phát các file âm thanh trong game.



Hình 10 Cửa sổ Audio Source

+ Audio clip: tham chiếu đến file âm thanh.

+ Mute: chơi ở chế độ như tắt tiếng.

+ Bypass effects: bộ lọc hiệu ứng áp dụng cho các nguồn âm thanh.

+ Bypass listener effects: Điều này là để nhanh chóng chuyển tất cả các hiệu ứng Listener on / off.

+ Play on awake: Nếu được kích hoạt, âm thanh sẽ bắt đầu chơi lúc scene được tải.

+ Loop: Kích hoạt tính năng này để làm cho Clip âm thanh lặp lại.

+ Pitch: Xác định ưu tiên của nguồn âm thanh này trong số tất cả những nguồn âm cùng tồn tại trong scene đó.

+ 3D Sound Setting: Cài đặt được áp dụng cho các nguồn âm thanh 3D.

+ 2D Sound Setting: Cài đặt được áp dụng cho các nguồn âm thanh 2D.

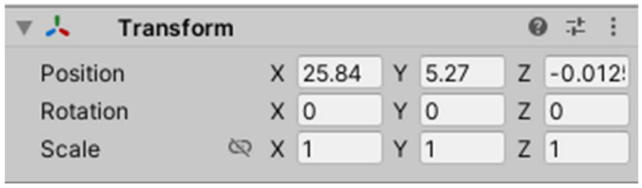
2.2.10 Camera

Camera trong Unity dùng để thể hiện khung hình, góc nhìn mà người chơi có thể nhìn thấy được trong game.

Trong một game có thể thiết lập nhiều camera để chia nhỏ màn hình người chơi, tạo các hiệu ứng, hoặc tùy chỉnh như nhìn được từ phía sau, bản đồ thu nhỏ, …

2.2.11 Transform

Quản lý vị trí, xoay, tỉ lệ của object.



Hình 11 Cửa sổ Transform

+ Position: vị trí hiện tại của object.

+ Rotation: độ xoay của object theo các trục x, y, z.

+ Scale: độ phóng to, thu nhỏ theo các trục x, y, z.

CHƯƠNG III: NỘI DUNG THỰC HIỆN

3.1 Kịch bản game

Người chơi sẽ đóng vai vào một anh chàng có tên là Virtual Guy và bắt đầu cuộc hành trình cuộc phiêu lưu của anh ta qua các màn chơi.

3.2 Xây dựng game

Game được xây dựng và lấy ý tưởng lỗi chơi từ tựa game Mario một trò chơi điện tử dưới dạng platform do Nintendo phát triển vào năm 1985.

3.2.1 Nhân vật di chuyển

```
public class PlayerMovement : MonoBehaviour
{
    private Rigidbody2D rb;
    private BoxCollider2D coll;
    private Animator anim;
    private SpriteRenderer sprite;
    //them double jump
    private bool doubleJump;

    [SerializeField] private LayerMask jumpableGround;

    private float dirX = 0f;
    [SerializeField]private float moveSpeed = 7f;
    [SerializeField] private float jumpForce = 14f;

    6 references
    private enum MovementSate { idle, running, jumping, falling}
    // Hieu ung am thanh
    [SerializeField] private AudioSource jumpSE;

    Unity Message | 0 references
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        coll = GetComponent<BoxCollider2D>();
        anim = GetComponent<Animator>();
        sprite = GetComponent<SpriteRenderer>();
    }
```

```
void Update()
{
    dirX = Input.GetAxis("Horizontal");
    rb.velocity = new Vector2(dirX * moveSpeed, rb.velocity.y);

    if(Input.GetButtonDown("Jump") && IsGrounded())
    {
        jumpSE.Play();
        rb.velocity = new Vector2(rb.velocity.x, jumpForce);
    }

    dirX = Input.GetAxis("Horizontal");
    rb.velocity = new Vector2(dirX * moveSpeed, rb.velocity.y);

    if (IsGrounded() && !Input.GetButton("Jump"))
    {
        doubleJump = false;
    }

    if (Input.GetButtonDown("Jump"))
    {
        if (IsGrounded() || doubleJump)
        {
            rb.velocity = new Vector2(rb.velocity.x, jumpForce);
            doubleJump = !doubleJump;
        }
    }
    UpdateAnimationUpdate();

    if (Input.GetKey(KeyCode.RightShift))
        if (m_TempCooldown <= 0)
        {
            Fire();
            m_TempCooldown = m_FiringCooldown;
        }
    m_TempCooldown -= Time.deltaTime;
}
```

- Cập nhật lại hoạt ảnh của nhân vật sau mỗi hành động:

```
private void UpdateAnimationUpdate()
{
    MovementSate state;
    if (dirX > 0f)
    {
        state = MovementSate.running;
        sprite.flipX = false;
    }
    else if (dirX < 0f)
    {
        state = MovementSate.running;
        sprite.flipX= true;
    }
    else
    {
        state = MovementSate.idle;
    }
    if(rb.velocity.y > 1f)
    {
        state = MovementSate.jumping;
    }
    if(rb.velocity.y < -.1f)
    {
        state = MovementSate.falling;
    }

    anim.SetInteger("state", (int)state);
}
```

```
private bool IsGrounded()
{
    return Physics2D.BoxCast(coll.bounds.center, coll.bounds.size, 0f, Vector2.down, .1f, jumpableGround);
}
```

3.2.2 Camera

- Thiết lập camera sẽ di chuyển theo nhân vật

```
public class CameraController : MonoBehaviour
{
    [SerializeField] private Transform player;
    public Vector3 offset;
    [Range(1, 10)]
    public float smoothFactor;
    public Vector3 minValue, maxValue;

    Unity Message | 0 references
    void FixedUpdate()
    {
        Follow();
    }
    1 reference
    void Follow()
    {
        //define min max x, y, z

        Vector3 playerPos = player.position + offset;
        //kiem tra vi tri co ra khoi rang buoc hay khong?
        //gioi han boi cac gia tri min max
        Vector3 boundPos = new Vector3(
            Mathf.Clamp(playerPos.x, minValue.x, maxValue.x),
            Mathf.Clamp(playerPos.y, minValue.y, maxValue.y),
            Mathf.Clamp(playerPos.z, minValue.z, maxValue.z)
            );
        Vector3 smoothPos = Vector3.Lerp(transform.position, boundPos, smoothFactor * Time.fixedDeltaTime);
        transform.position = smoothPos;
    }
}
```

3.2.3 Máu nhân vật

- Thanh máu của nhân vật:

```
public class HealthBar : MonoBehaviour
{
    [SerializeField] private Health playerHealth;
    [SerializeField] private Image totalhealthBar;
    [SerializeField] private Image currenthealthBar;
    [SerializeField] private Image healthBar;    Tab to accept
    Unity Message | 0 references
    private void Start()
    {
        currenthealthBar.fillAmount = playerHealth.currentHealth / 10;
    }
    Unity Message | 0 references
    private void Update()
    {
        currenthealthBar.fillAmount = playerHealth.currentHealth / 10;
    }
}
```

- Sẽ cộng thêm máu cho nhân vật sau khi nhân vật "ăn" được mỗi trái tim.

```
public class HeartCollectible : MonoBehaviour
{
    [SerializeField] private float healthValue;
    Unity Message | 0 references
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.tag == "Player")
        {
            collision.GetComponent<Health>().addHealth(healthValue);
            gameObject.SetActive(false);
        }
    }
}
```

- Trừ máu khi nhân vật chạm phải bẫy

```
public void TakeDamage(float _damage)
{
    currentHealth = Mathf.Clamp(currentHealth - _damage, 0, startingHealth);
    if(currentHealth > 0)
    {
        anim.SetTrigger("hurt");
        hurtSE.Play();
    }
    else
    {
        if (!dead)
        {
            anim.SetTrigger("death");
            deathSE.Play();
            rb.bodyType = RigidbodyType2D.Static;
            GetComponent<PlayerMovement>().enabled = false;
            dead = true;
        }
    }
}
```

3.2.4 Sát thương do bẫy

```
public class EnemyDamage : MonoBehaviour
{
    [SerializeField] protected float damage;

    protected void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "Player")
        {
            collision.GetComponent<Health>().TakeDamage(damage);
        }
    }
}
```

3.2.5 Xây dựng Jump Pad

- Là vị trí giúp nhân vật nhảy cao hơn.

```
public class JumpPad : MonoBehaviour
{
    [SerializeField] private float do_nay = 20f;
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            collision.gameObject.GetComponent<Rigidbody2D>().AddForce(Vector2.up * do_nay, ForceMode2D.Impulse);
        }
    }
}
```

3.2.6 Xây dựng GamePlayManager

Chia nhỏ các GameState

```
public enum GameState {
    Home,
    GamePlay,
    Pause,
    GameOver
}
```

Khai báo các trường cho GameState

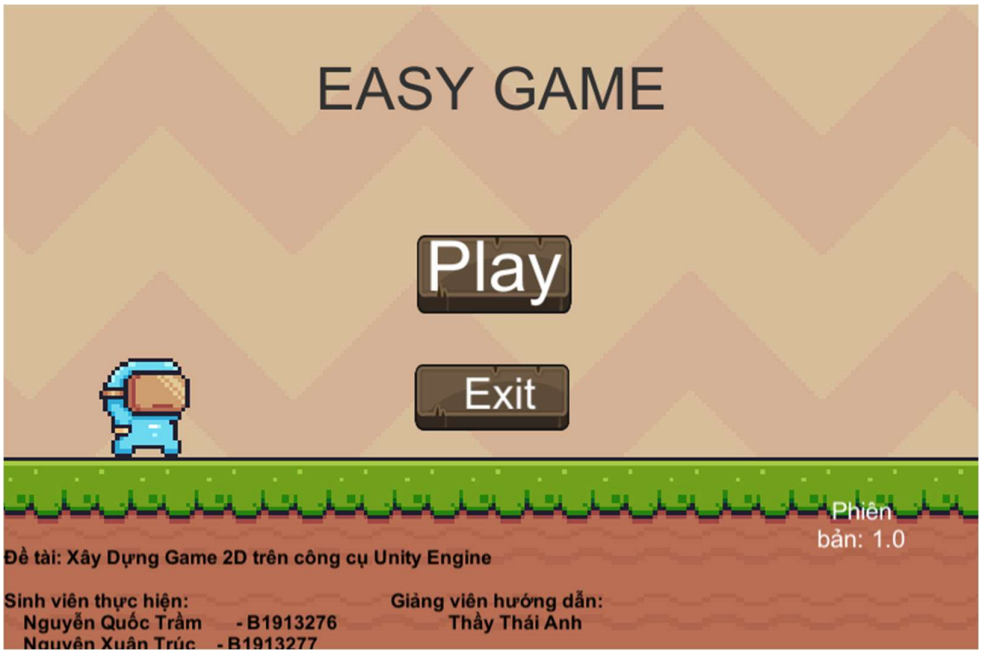
```
[SerializeField] HomePanel m_Home;
[SerializeField] GamePlayPanel m_GamePlayPanel;
[SerializeField] GameOverPanel m_GameOverPanel;
[SerializeField] PausePanel m_PausePanel;

private GameState m_GameState;
```

SetActive cho các State

```
private void Start()
{
    m_Home.gameObject.SetActive(false);
    m_GamePlayPanel.gameObject.SetActive(false);
    m_PausePanel.gameObject.SetActive(false);
    m_GameOverPanel.gameObject.SetActive(false);
    SetState(GameState.GamePlay);
}
private void SetState(GameState state)
{
    m_GameState = state;
    m_Home.gameObject.SetActive(m_GameState == GameState.Home);
    m_GamePlayPanel.gameObject.SetActive(m_GameState == GameState.GamePlay);
    m_PausePanel.gameObject.SetActive(m_GameState == GameState.Pause);
    m_GameOverPanel.gameObject.SetActive(m_GameState == GameState.GameOver);
    if (m_GameState == GameState.Pause)
        Time.timeScale = 0;
    else
        Time.timeScale = 1;
}
```

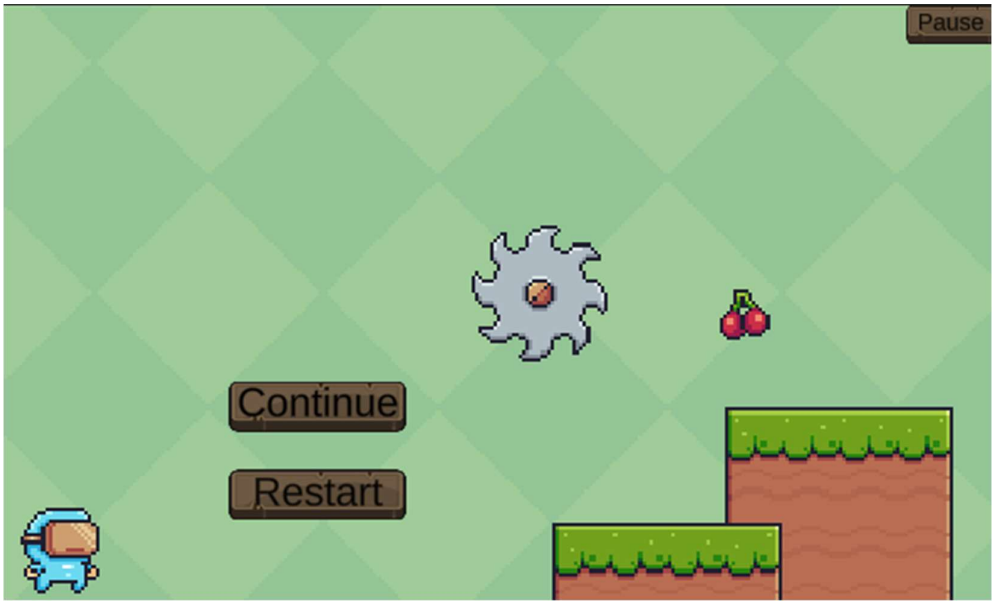
3.3 Demo game

3.3.1 Màn hình Start



Hình 12 Màn hình Start

3.3.2 Màn hình game play



Hình 13 Màn hình game play

CHƯƠNG IV: TỔNG KẾT

4.1 Kết quả đạt được

- Hiểu rõ được tổng quan về công cụ Unity.
- Hiểu rõ được tác dụng của các chức năng cơ bản trong Unity.
- Nắm được cách làm game với Unity.
- Xây dựng được trò chơi.

4.2 Hạn chế của đề tài

- Còn nhiều chức năng chưa hoàn thiện.
- Chưa bắt được hết các lỗi của hệ thống

4.3 Hướng phát triển của đề tài

- Tiếp tục nghiên cứu nhằm tối ưu trò chơi, sửa chữa những lỗi còn tồn đọng.

- Phát triển thêm các chức năng, tùy chọn để trò chơi hấp dẫn hơn.

4.4 Kinh nghiệm rút ra

- Cần phải có sự đầu tư nhiều hơn nữa để có thể tạo ra sản phẩm tốt nhất.
- Phải biết sắp xếp thời gian 1 cách hợp lý để tránh những khó khăn.

Tài liệu tham khảo

[1] Unity Manual

- Tác giả: Unity Technologies
- Nội dung: Cung cấp các định nghĩa, cách thức sử dụng các chức năng của Unity.
- Đường dẫn: https://docs.unity3d.com/Manual

[2]. Build a 2D Platformer Game in Unity | Unity Beginner Tutorial

- Tác giả: Coding in Flow
- Nội dung: Cung cấp ý tưởng, code tham khảo.
- Đường dẫn: https://www.youtube.com/@codinginflow

[3]. Unity 2D Platformer for Complete Beginners

- Tác giả: Pandemonium
- Nội dung: Cung cấp ý tưởng, code tham khảo.
- Đường dẫn: https://www.youtube.com/@PandemoniumGameDev