TRƯỜNG ĐẠI HỌC BÁCH KHOA

THÀNH PHỐ HỒ CHÍ MINH

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Giảng viên hướng dẫn: Mai Đức Trung

Thành viên nhóm và công việc phân công

Họ và Tên	MSSV	Công việc thực hiện	Phần trăm công việc
Bùi Bá Anh	1710447	Hiện thực nhân vật cùng với các tính năng	20%
Vương Anh Khoa	1711803	Kiểm tra các chức năng và hiệu chỉnh thành phẩm	20%
Cáp Đặng Xuân Kiệt	1711852	Hiện thực scene, button, text làm báo cáo và slide	20%
Nguyễn Thành Thông	1710313	Hiện thực scene, button, text và tổng hợp các thành phần	20%
Trần Hữu Trí	1713658	Tạo đối tượng rơi trong game, audio và xử lý đụng độ	20%

1. Giới thiệu tổng quan về game

1.1 Mô tả game

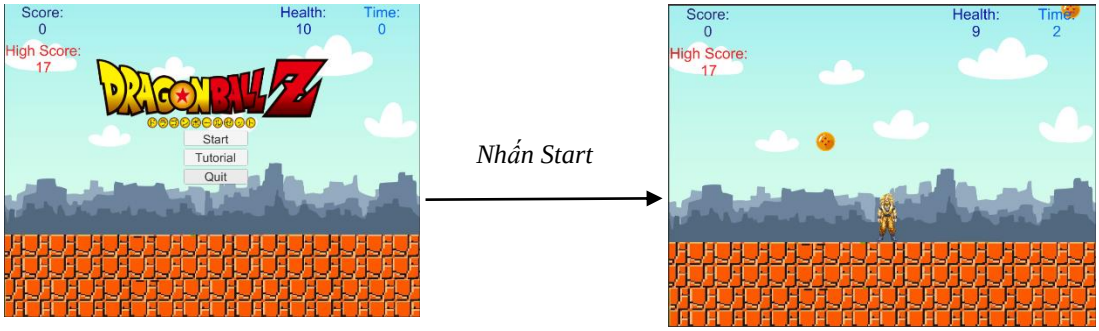
HỨNG HOA QUẢ là một dòng game không hề xa lạ với tất cả chúng ta. Đây là một game với cách chơi khá đơn giản nhưng rất được các bạn trẻ yêu thích. Nhiệm vụ của người chơi là điều khiển một đối tượng (có thể là giỏ hoa quả, nhân vật truyện tranh,..) di chuyển qua lại để hứng những vật thể rơi từ phía trên của màn hình game. Bên cạnh đó, một vài vật thể lạ (hứng sẽ bị trừ mạng, trừ điểm, trừ thời gian,…) có thể được thiết lập để tăng độ khó cho game.

Sản phẩm của nhóm cũng là một game thuộc nhóm game này với cách chơi tương tự, tuy nhiên nhân vật trong game sẽ không đơn giản chỉ là những đối tượng tĩnh. Nhóm đã xây dựng nhân vật có chuyển động linh hoạt hơn và thực hơn để thu hút người chơi. Bên cạnh đó, những trái cây cũng sẽ được thay bằng hình ảnh khác để phù hợp với tạo hình nhân vật. Tính năng lưu điểm cao nhất (highscore) cũng được thêm vào game.

1.2 Scene trong game

Scene vào game:

Scene chính:



1.3 Hướng dẫn chơi

Người chơi sẽ có 5 mạng lúc bắt đầu game, nhiệm vụ của họ sẽ là điều khiển nhân vật ăn tất cả BÓNG (ball) rơi ra, nếu để hụt một viên thì bị trừ một mạng. Trò chơi sẽ kết thúc khi người chơi để mất cả 5 mạng.

Bên cạnh đó, ngoài ball, sẽ có TRÁI TIM (heart), BOOM (bomb), VIÊN NĂNG LƯỢNG (energy), PHI TIÊU (shuriken) và khối GENKI DAMA (genki) được thả rơi đồng thời với ball. Một heart sẽ ứng với một mạng cộng thêm, những đối tượng còn lại đều trừ điểm và mạng:

- 1 điểm và 1 mạng với bomb
- 2 điểm 2 mạng với shuriken
- 3 điểm 3 mạng với energy
- 20 điểm 20 mạng với genki

Thời gian xuất hiện đối với các đối tượng sẽ tỉ lệ thuận với số điểm mà nó mang lại hoặc mang đi.

Một điểm đáng lưu ý khác là tất cả các đối tượng rơi sẽ được thả rơi với tần suất lớn hơn sao thông báo chuyển level.

Trong lúc chơi, đôi lúc sẽ có một dòng thông báo WARNING hiện ra, điều này nghĩa là tất cả các vật thể rơi chuẩn bị độ xuống với tần số cực lớn.

Các phím được sử dụng để điều khiển trong game:

- Phím mũi tên trái: điều khiển nhân vật sang trái
- Phím mũi tên phải: điều khiển nhân vật sang phải
- Phím mũi tên lên: nhảy

1.4 Tạo hình nhân vật

Nhân vật hứng ở trong game là SonGoku, một trong những nhân vật truyện tranh nổi tiếng nhất đối với giới trẻ hiện nay.

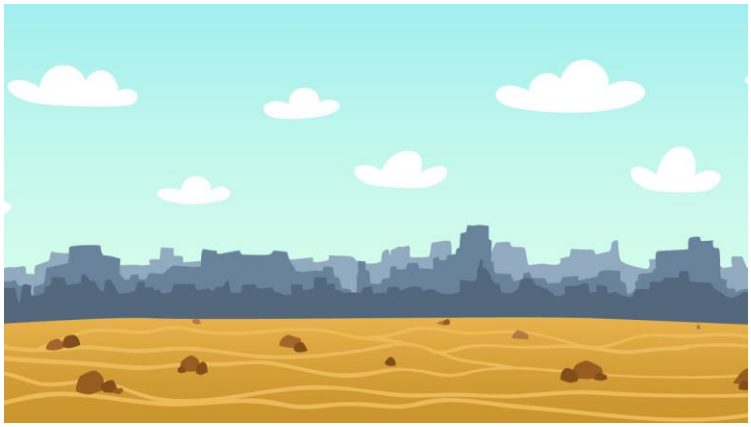
Tạo hình của Goku và những hình ảnh được sử dụng để thiết kế animation

1.5 Tạo hình đối tượng khác trong game

1.5.1 Cảnh vật nền

Background vào game sẽ bao gồm những thành phần sau:

- Phong nền



- Logo game

- Tường

1.5.2 Đối tượng rơi

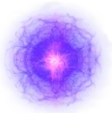
- Ball

- Heart

- Bomb

- Shuriken

- Energy



- Genki

1.5.3 Thông báo WARNING

1.5.4 Text

Những text trong game đều là phong mặc định của unity với kiểu chữ Arial. Kích cỡ đều là 36 và màu như sau:

- Score
- Health
- Time
- High Score
- Tutorial
- Level

1.5.5 Button

Tương tự text, các button đều là button mặc định có sẵn trong unity. Bên dưới là một ví dụ về button trong game:

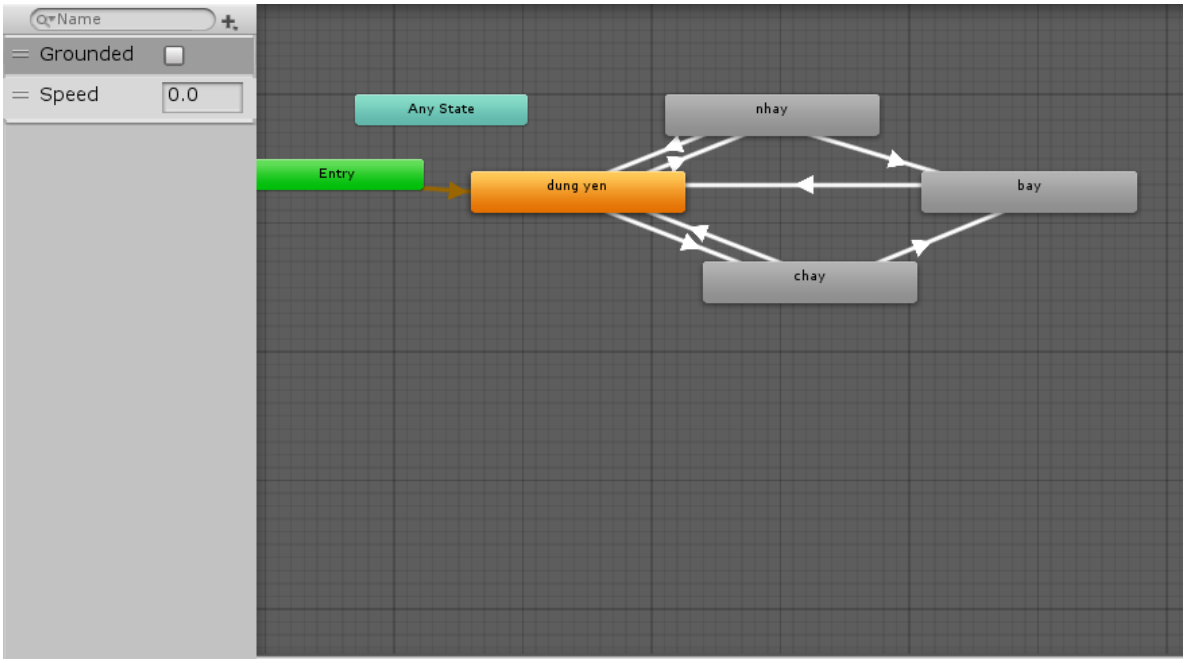
Nút Home được sử dụng để quay về trang chính

2. Hiện thực các tính năng trong game

2.1 Những tính năng cơ bản

2.1.1 Điều khiển nhân vật

Trước hết, ta sử dụng những hình ảnh nhân vật đã đề cập ở trên để tạo animation cho nhân vật. Vì có 4 hoạt động là đứng yên, chạy, nhảy và bay nên ta cần 4 animation tương ứng. Sau đó nối animator cho Goku và tạo hai parameter (1 bool và 1 float)



Ở phần hiện thực code, ta sẽ khai báo các biến toàn cục như sau:

- Tốc độ tức thời speed (float)
- Tốc độ tối đa maxSpeed (float)
- Tốc độ lúc nhảy jump (float)
- Chạm đất grounded (bool)
- Hướng hiện tại của Goku faceright (bool)

Hai component cũng sẽ được thêm vào là Rigidbody2D (gọi là r2) và Animator (gọi là anim). R2 sẽ lưu các thuộc tính vật lý (ở đây là lực) và anim lưu các animation được tạo ban đầu.

Tiếp theo, ta hiện thực các hàm, bao gồm:

- Hàm Update() : Được gọi một lần mỗi frame, đây là hàm điều khiển Goku nhảy lên khi nhấn nút UpArrow. Hàm được hiện thực như sau:

```
// Update is called once per frame
void Update()
{
    if (Input.GetKeyDown(KeyCode.UpArrow))
    {
        r2.AddForce(Vector2.up * jump); // Khi nút UpArrow được nhấn, một lực theo phương đứng sẽ được cấp cho Goku
    }
    anim.SetBool("Grounded", grounded); // đặt lại giá trị parameter Grounded của anim
    anim.SetFloat("Speed", Mathf.Abs(r2.velocity.x)); // đặt lại giá trị parameter speed của anim ( bằng với vận tốc ngang hiện thời)
}
```

- Hàm FixedUpdate() : Đây là hàm điều khiển Goku di chuyển trái phải. Hàm được hiện thực như sau:

```
private void FixedUpdate()
{
    float h = Input.GetAxis("Horizontal"); // hướng quay đến của Goku, hướng dương là 1, âm là -1
    r2.AddForce((Vector2.right) * speed * h); // kích lực cho Goku theo hướng của h
    if (r2.velocity.x > maxSpeed)
        r2.velocity = new Vector2(maxSpeed, r2.position.y); // Xét điều kiện về vận tốc của Goku trong trường hợp vận tốc dương, nếu lớn hơn maxSpeed thì đặt về bằng maxSpeed
    if (r2.velocity.x < -maxSpeed)
        r2.velocity = new Vector2(-maxSpeed, r2.position.y);// Xét điều kiện về vận tốc của Goku trong trường hợp vận tốc âm, nếu lớn hơn maxSpeed thì đặt về bằng maxSpeed
    if (h > 0 && !faceright)
        Flip(); // gọi hàm Flip() nếu giá trị của faceright nếu khác với h
    if (h < 0 && faceright)
        Flip();// gọi hàm Flip() nếu giá trị của faceright nếu khác với h
}
```

- Hàm Flip() : Chức năng chính của hàm là thay đổi các giá trị về hướng hiện tại của Goku.

```
public void Flip()
{
    faceright = !faceright; // đảo giá trị của faceright
    Vector3 Scale;
    Scale = transform.localScale; // biến scale bằng localScale của Goku
    Scale.x *= -1; // đảo giá trị x của scale
    transform.localScale = Scale;// lưu lại vào localScale
}
```

2.1.2 Sinh các vật thể rơi ngẫu nhiên

2.1.2.1 Thông báo tăng tần xuất rơi một vài đối tượng

Kể từ lúc bắt đầu chơi cho đến một khoảng thời cụ thể (không cố định, thay đổi theo từng đợt chơi), các vật thể rơi sẽ tăng dần tần suất rơi và thông báo chuyển level sẽ hiện lên.

Việc này sẽ được thực hiện bởi lệnh *yield return new WaitForSeconds().*

Thông báo WARNING cũng được thực hiện bởi cách tương tự.

2.1.2.2 Sinh ball

Để sinh ra ball ở một vị trí ngẫu nhiên , trước tiên ta khai báo hai biến cục bộ là spawnRotation và spawnPosition

```
Quaternion spawnRotation = Quaternion.identity; // tạo góc quay
Vector3 spawnPosition = new Vector3(Random.Range(-maxWidth, maxWidth), transform.position.y, 0.0f); // tạo một biến vị trí với x ngẫu nhiên
```

Việc cần làm tiếp theo chỉ là sinh ball theo góc và vị trí đã khai báo

```
Instantiate(ball, spawnPosition, spawnRotation); // sinh ball với vị trí và góc quay như trên
```

Đối với ball, vì không có điều kiện rằng buộc nên nó sẽ được sinh ra theo thời gian cố định.

2.1.2.3 Sinh các vật thể khác

Việc sinh ra các vật thể còn lại giống với ball, tuy nhiên sẽ có một biến ngẫu nhiên để kiểm soát việc này với những khoảng chọn khác nhau.

Ví dụ với heart, nhóm đã tạo ra một biến chạy trong khoảng từ 0-5, nếu nó bằng 0 thì heart mới được sinh ra

```
i = Random.Range(0, 5);

if (i == 0)
{
    yield return new WaitForSeconds(Random.Range(0.7f, 1.5f));
    Instantiate(heart, spawnPosition1, spawnRotation);
}
```

2.1.3 Xử lý va cham

2.1.3.1 Xóa vật khi va cham với Wall

```
void OnCollisionEnter2D(Collision2D other) // Xác định va cham với wall
{

    if (other.gameObject.tag == "Ball" || other.gameObject.tag == "Heart" || other.gameObject.tag == "Boom"|| other.gameObject.tag=="Energy"|| other.gameObject.tag=="Shuriken"||other.gameObject.tag=="GenkiDama")
    {
        Destroy(other.gameObject); //Nếu va chạm với bất kỳ vật thể nào -> xóa

        //khi bóng rơi xuống đất thì mất điểm
        if (other.gameObject.tag == "Ball")
            if(gamobj.health>0)
                gamobj.health--;
        //cập nhật điểm
        desobj.UpdateScore();
        //cập nhật health
        gamobj.UpdateHealthText();
    }
}
```

2.1.3.2 Xóa vật khi va chạm với Goku

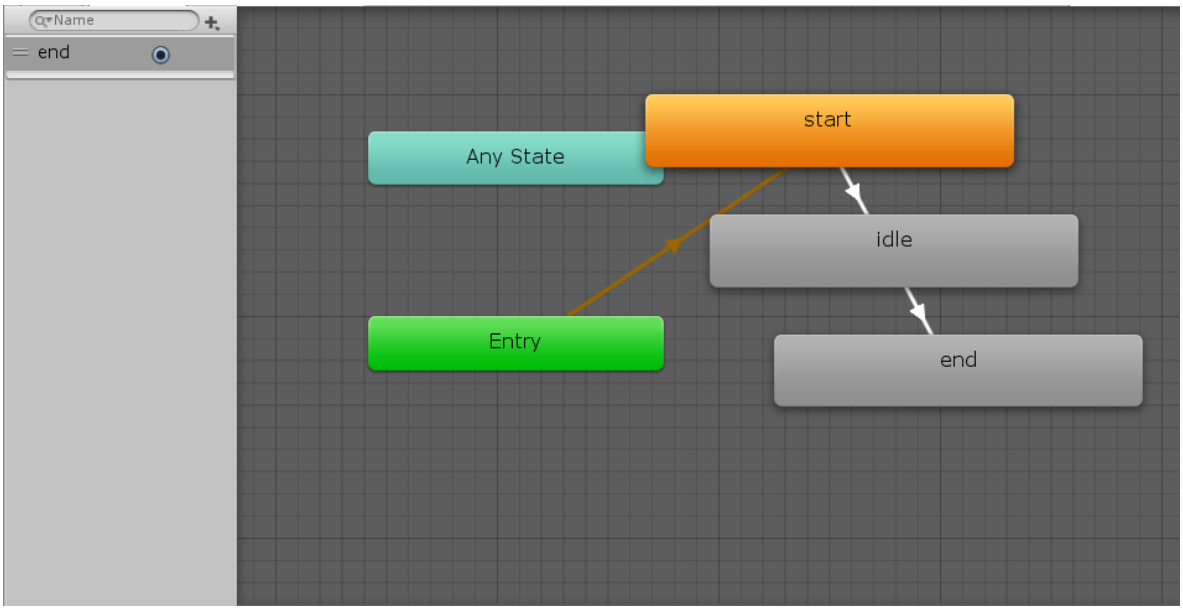
```
void OnCollisionEnter2D(Collision2D other) // Hàm xác định va chạm
{
    //tăng điểm khi ăn bóng
    if (other.gameObject.tag == "Ball")
    {
        //Nếu va chạm với banh -> xóa banh
        Destroy(other.gameObject);
        AudioControl.PlaySound(soundsGame.ball); // chạy audio
        score += 1; // cộng điểm
        UpdateScore(); // cập nhật điểm
    }
```

Trên đây là hàm xử lý va chạm với ball, các vật thể khác đều được thực hiện tương tự.

2.2 Các tính năng phụ

2.2.1 Hiện scene và chuyển scene

Trước tiên, phải tạo một panel gắn với scene cần chuyển đồng thời tạo ba animation gắn với panel đó, bao gồm start (chuyển từ màn hình đen sang scene cần hiện), idle (giữ nguyên scene) và end (chuyển từ scene sang màn hình đen và chuyển scene). Nối animator cho panel và tạo một parameter trigger.



Giải thuật được hiện thực như sau

```
public class SceneTransitions : MonoBehaviour {

    public Animator transitionAnim;// panel
    public string sceneName; // chứa tên của scene mới muốn chuyển đến
    public AudioClip soundIntro; // audio
    public static AudioSource instance;
    void Start()
    {
        instance = GetComponent<AudioSource>();
        instance.PlayOneShot(soundIntro); // Load audio chuyển scene
    }
    void Update()
    {
        StartCoroutine(Wait());
        StartCoroutine(LoadScene());

    }

    IEnumerator LoadScene()
    {

        transitionAnim.SetTrigger("end");
        yield return new WaitForSeconds(2.4f); // chờ trong 2.4s
        SceneManager.LoadScene(sceneName); // chuyển sang scene mới
    }
    IEnumerator Wait()
    {
        yield return new WaitForSeconds(2.2f); // chờ trong 2.2s
    }
}
```

2.2.2 Audio

Trong game sẽ có 4 loại âm thanh được phát ra bao gồm :

- Nhạc nền
- Nhạc va chạm ball
- Nhạc va chạm heart
- Nhạc va chạm các vật thể khác

Để phát ra từng loại nhạc với từng thời điểm và loại va chạm tương ứng, trước tiên nhóm đã khai báo 4 biến Audioclip tương ứng với 4 audio và 1 biến Audiocontrol.

```
public AudioClip soundBall;
public AudioClip soundHeart;
public AudioClip soundBoom;
public AudioClip soundNhacnen;
public static AudioControl instance;
```

Instance sẽ điều khiển xem âm thanh nào sẽ được phát vì vậy nên nó phải được cấp giá trị của component Audio Control

```
void Start() {
    instance = this;
    instance.GetComponent<AudioSource>().volume = 0.14f;
}
```

Cuối cùng, hàm xét điều kiện để phát một âm thanh sẽ được dựng lên như sau

```
public static void PlaySound(soundsGame currentSound)
{
    switch (currentSound)
    {
        case soundsGame.ball:
            {
                instance.GetComponent<AudioSource>().PlayOneShot(instance.soundBall);
            }
            break;
        case soundsGame.heart:
            {
                instance.GetComponent<AudioSource>().PlayOneShot(instance.soundHeart);
            }
            break;
        case soundsGame.boom:
            {
                instance.GetComponent<AudioSource>().PlayOneShot(instance.soundBoom);
            }
            break;
        case soundsGame.nhacnen:
            {
                instance.GetComponent<AudioSource>().PlayOneShot(instance.soundNhacnen);
            }
            break;
    }
}
```

2.2.3 High score

Đối với tính năng nay, ta mượn có thể mượn dùng lại biến score ở trên phần 2.1.3.2 đồng thời khai báo hai biến mới là highscore (int, giá trị là điểm lớn nhất) và highscoreText (text, giá trị là string xuất ra màn hình)

Việc hiện thực giải thuật khá đơn giản như sau:

```
if (score > highscore)
{
    highscore = score; // nếu score>highscore thì thay đổi giá trị của highscore
    PlayerPrefs.SetInt("highscore", highscore);
    highscoreText.text = "Highscore:\n" + highscore; // xuất ra màn hình
}
```

2.2.4 Tính năng với button

2.2.4.1 Button Start

Đây là button được hiển thị trong scene chính, với chức năng là vô hiệu hóa các đối tượng không cần thiết trên màn hình (logo, các nút nhấn khác), kích hoạt Goku và khởi động việc sinh các đối tượng rơi.

Việc hiện thực hàm gồm sử dụng các lệnh Gameobject.Setactive để vô hiệu hoặc kích hoạt các đối tượng, gọi hàm điều khiển Goku và hàm sinh ngẫu nhiên đã được nêu ở trên

```
public void StartGame()
{
    //Tắt "Dragon Ball Z" +  button
    splashScreen.SetActive(false);
    startButton.SetActive(false);
    TutButton.SetActive(false);
    QuitButton.SetActive(false);
    Goku.SetActive(true);
    timeLeft = 0;
    UpdateText();
    //Spawn bóng
    StartCoroutine(Spawn());
}
```

2.2.4.2 Button Tutorial và Home

Tutotial là button được hiện thị trong scene chính và home là button đươc hiển thị sau khi nhấn tutorial. Hai button này điều khiển việc vô hiệu hoặc kích hoạt các đối tượng trong game

```
//Điều khiển Tutorial button
public void Tutorial()
{
    splashScreen.SetActive(false);
    startButton.SetActive(false);
    TutButton.SetActive(false);
    Goku.SetActive(false);
    QuitButton.SetActive(false);
    TutText.SetActive(true);
    HomeButton.SetActive(true);
}
```

```
// Điều khiểu home button
public void Home()
{
    splashScreen.SetActive(true);
    startButton.SetActive(true);
    TutButton.SetActive(true);
    QuitButton.SetActive(true);
    Goku.SetActive(false);
    TutText.SetActive(false);
    HomeButton.SetActive(false);
}
```

2.2.4.3 Button Restart

Button này chỉ được hiển thị khi gameover, nó sẽ gọi hàm để load lại scene chính.

```
//Restart lại game
public void RestartGame()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
}
```

2.2.4.4 Button Quit

Đây là button duy nhất hiển thị ở hai nơi trong game (ở scene chính và khi gameover), nó sẽ gọi hàm Appication.Quit() để thoát game.

3. Class Diagram

4. Nhận xét

4.1 Mục tiêu đã đạt được

Về animation, nhóm đã thiết kế thành công nhân vật Goku với những chuyển động tinh tế và khá "thực" và cách tự động chuyển scene sáng tạo.

Về tính năng, nhóm đã hoàn thành được gần như đầy đủ các tính năng cần phải có của dòng game kinh điển này, bên cạnh đó còn phát triển thêm một vài tính năng phụ khác (highscore, level,..).

Về đồ họa, những hình ảnh trong game đều được thành viên chọn lựa kĩ càng và chia cắt tỉ mỉ (khi làm animation) khiến đồ họa tuy không quá xuất sắc nhưng cũng đủ để nổi trội.

4.2 Khó khăn

Mặt khách quan, Unity không phải là một game engine quá dễ đối với những người mới bắt đầu làm game. Với hệ thống các hàm, thư viện,… khổng lồ nên việc xử dụng thích hợp là một điều khá khó khăn. Bên cạnh đó việc dùng C# để lập trình Script cũng tạo ra không ít khó khăn với những người không quen viết bằng ngôn ngữ này.

Mặt chủ quan, do sinh viên vẫn còn khá lạ lẫm với Unity nên tốn nhiều thời gian mò mẫm làm chậm tiến độ của bài tập lớn khá nhiều.

4.3 Hướng phát triển

- **Về đồ họa:** Tuy đồ họa trong game có phần khá nổi trội, tuy nhiên những hình ảnh này lại không mang lại sự mới lạ cho người chơi. Trong tương lai có thể nhóm sẽ tự thiết kế lại hình ảnh cho game.
- **Về tính năng:** Có một vài tính năng nhóm vẫn chưa kịp hoàn thành khi tới deadline như xoay shuriken lúc đang rơi, thả rơi 7 loại ball với 7 mức điểm khác nhau,… Đây cũng là những tiếc nuối khá lớn mà nhóm hi vọng có thể sửa chữa trong tương lai.
- **Về ý tưởng:** Nhóm có ý tưởng sẽ làm thêm một game phụ trong game này để tăng sự đa dạng cũng như thu hút thêm người chơi