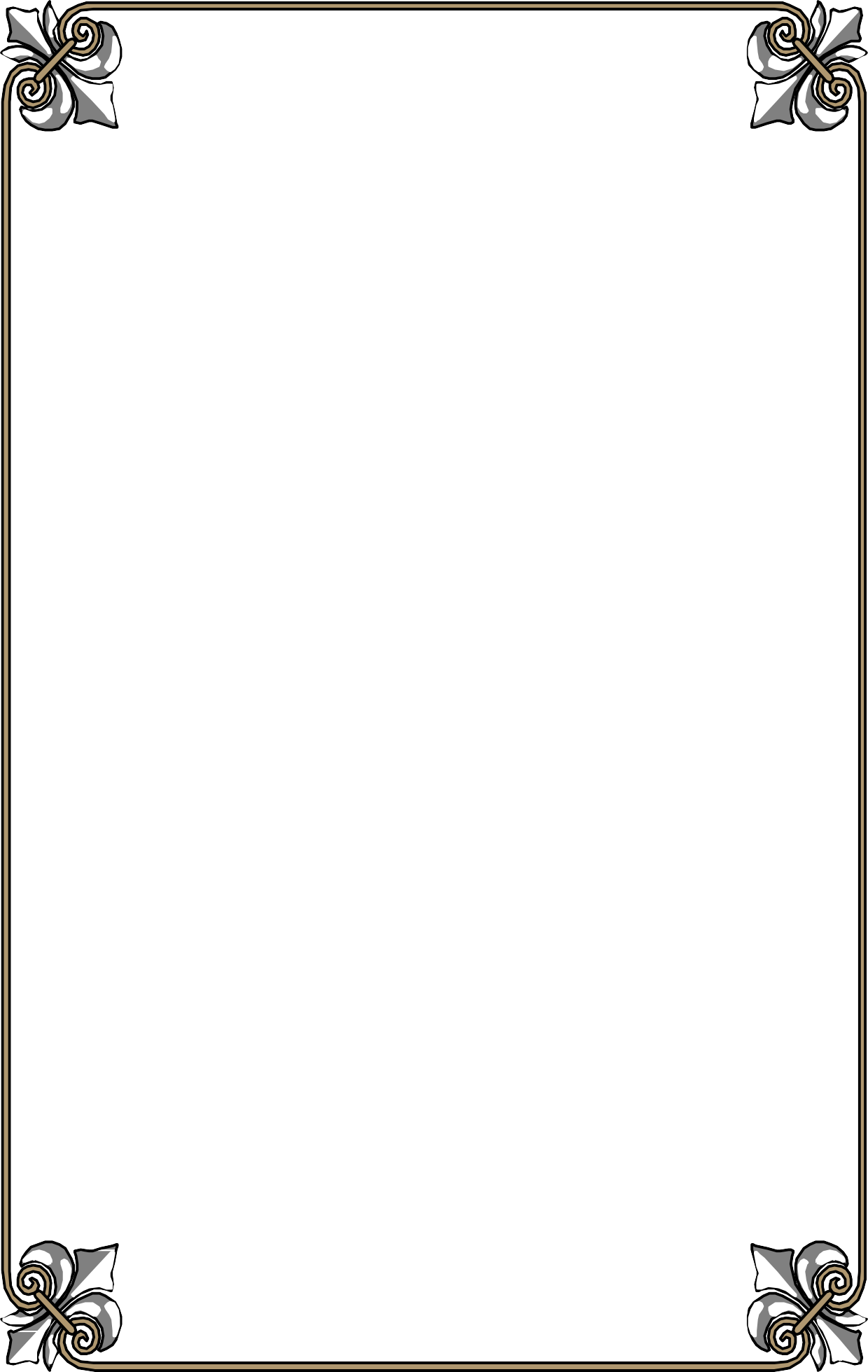
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**



**VIỆN ĐIỆN TỬ-VIỄN THÔNG**

**BÁO CÁO CÁ NHÂN BÀI TẬP DÀI MÔN HỌC**

**Kỹ thuật lập trình nội dung số**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | TS. Phạm Văn Tiến |
| Sinh viên thực hiện: | MSSV |
| ***Lưu Đình Việt Ân*** | ***20186300*** |
|  |  |

*Hà Nội, 1/2023*

Các công việc đã thực hiện.

1. Tải và cài đặt Visual Studio.

* Truy cập vào trang chủ của Visual Studio tại địa chỉ https://visualstudio.microsoft.com/
* Chọn phiên bản của Visual Studio mà bạn muốn tải về và cài đặt. Có nhiều phiên bản khác nhau, bao gồm Visual Studio Community, Visual Studio Professional và Visual Studio Enterprise.
* Nhấp vào nút "Tải xuống" để bắt đầu quá trình tải về của phần mềm.
* Khi tải về hoàn tất, mở tập tin cài đặt và chạy các bước cài đặt để cài đặt Visual Studio.

1. Tải và cài đặt Unity Hub. Bổ sung gói.

* Truy cập trang web của Unity Hub: https://unity.com/hub.
* Nhấn vào nút "Download Unity Hub" để tải về phần mềm.
* Mở tập tin tải về và chạy các bước cài đặt theo hướng dẫn trên màn hình.
* Sau khi cài đặt xong, mở Unity Hub và đăng nhập bằng tài khoản Unity của bạn hoặc tạo tài khoản mới.

Để tải thêm Addon Package, thực hiện các bước sau:

* Trong Unity Hub, nhấn vào nút "Installs" để mở danh sách các phiên bản Unity đã cài đặt.
* Chọn phiên bản Unity muốn tải Addon Package và nhấn vào nút "Add Modules" tại phần "Modules".
* Chọn "Universal Windows Platform" từ danh sách các Addon Package và nhấn nút "Apply".
* Chờ cho quá trình tải và cài đặt hoàn tất.

1. Tải và cài đặt Blender.
2. Tải và cài đặt gói C# trên Visual Studio.
3. Tải và cài đặt Java Open Street Map.
4. Tìm kiếm, cấu hình và chỉnh sửa các đối tượng sử dụng trong phối cảnh.

Đối tượng chính:

* Prefabs và scene: <https://drive.google.com/file/d/1QM_mz50JA2be1G279foCSXs8pJhrHizm/view>
* Dotween engine:

<https://assetstore.unity.com/packages/tools/animation/dotween-hotween-v2-27676>

* Klak motion:

<https://github.com/keijiro/Klak>

Các đối tượng phụ:

* Alien: <https://www.turbosquid.com/3d-models/free-3ds-mode-alien/400453>
* Dragan: <https://www.turbosquid.com/3d-models/3d-dragan-bathroom-set-1991893>
* Balloon: <https://www.turbosquid.com/3d-models/free-3ds-mode-balloon-red-string/479788>

1. Xây dựng phối cảnh chính.
2. Thiết kế logic, các quy tắc vật lý của đối tượng chính.
3. Xây dựng và đóng gói API của đối tượng chính.
4. BladesController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class BladesController : MonoBehaviour

{

public enum Axis

{

x,

y,

z

}

public Axis rotationAxis;

private float bladeSpeed;

public float BladeSpeed

{

get

{

return bladeSpeed;

}

set

{

bladeSpeed = Mathf.Clamp(value,0,3000);

}

}

public bool inverseRotation = false;

private Vector3 Rotation;

float rotateDegree;

// Start is called before the first frame update

void Start()

{

Rotation = transform.localEulerAngles;

}

// Update is called once per frame

void Update()

{

if (inverseRotation)

rotateDegree -= bladeSpeed \* Time.deltaTime;

else

rotateDegree += bladeSpeed \* Time.deltaTime;

rotateDegree = rotateDegree % 360;

switch(rotationAxis)

{

case Axis.y:

transform.localRotation = Quaternion.Euler(Rotation.x, rotateDegree, Rotation.z);

break;

case Axis.z:

transform.localRotation = Quaternion.Euler(Rotation.x, Rotation.y, rotateDegree);

break;

default:

transform.localRotation = Quaternion.Euler(rotateDegree, Rotation.y, Rotation.z);

break;

}

}

}

1. HelicopterMainEngines.cs

using System.Collections;

using System.Collections.Generic;

using System.IO;

using UnityEngine;

using DG.Tweening;

using UnityEngine.Events;

public class HelicopterMainEngine : MonoBehaviour

{

Rigidbody helicopterRigid;

public BladesController MainBlade;

public BladesController SubBlade;

private float enginePower;

public float EnginePower

{

get

{

return enginePower;

}

set

{

MainBlade.BladeSpeed = value \* 250;

SubBlade.BladeSpeed = value \* 500;

enginePower = value;

}

}

public float effectiveHeight;

public float engineStartSpeed;

public float EngineLift = 0.01f;

public float ForwardForce;

public float BackwardForce;

public float TurnForce;

private float TurnForceHelper = 1.5f;

public float ForwardTiltForce;

public float TurnTiltForce;

private Vector2 Movement = Vector2.zero;

private Vector2 TILTING = Vector2.zero;

public LayerMask groundLayer;

private float distanceToGround;

//set true for using HandleGround function

public bool isOnGround = true;

private float turning = 0f;

public UnityEvent OnTakeOff;

public UnityEvent Onland;

bool isFirstTime;

// Start is called before the first frame update

void Start()

{

helicopterRigid = GetComponent<Rigidbody>();

}

// Update is called once per frame

void Update()

{

HandleGroundCheck();

HandleInput();

HandleInvoks();

HandleEngine();

}

protected void FixedUpdate()

{

HelicopterHover();

HelicopterMovements();

HelicopterTilting();

}

void HandleInput()

{

if (!isOnGround)

{

Movement.x = Input.GetAxis("Horizontal");

Movement.y = Input.GetAxis("Vertical");

if (Input.GetKey(KeyCode.C))

{

EnginePower -= EngineLift;

if (EnginePower < 0)

{

EnginePower = 0;

}

}

}

/\* Movement.x = Input.GetAxis("Horizontal");

Movement.y = Input.GetAxis("Vertical");

if (Input.GetKey(KeyCode.C))

{

EnginePower -= EngineLift;

if (EnginePower < 0)

{

EnginePower = 0;

}

}\*/

if (Input.GetAxis("Throttle") > 0)

{

EnginePower += EngineLift;

}

else if (Input.GetAxis("Vertical") > 0 && isOnGround)

{

EnginePower = Mathf.Lerp(EnginePower, 17.5f, 0.003f);

}

else if (Input.GetAxis("Throttle") < 0.5f && !isOnGround)

{

EnginePower = Mathf.Lerp(EnginePower, 10f, 0.003f);

}

}

void HandleInvoks()

{

if (!isOnGround && isFirstTime)

{

OnTakeOff.Invoke();

isFirstTime= false;

} else if (isOnGround && !! !isFirstTime)

{

Onland.Invoke();

isFirstTime= true;

}

}

void HandleGroundCheck()

{

RaycastHit hit;

Vector3 direction = transform.TransformDirection(Vector3.down);

Ray ray = new Ray(transform.position, direction);

if (Physics.Raycast(ray, out hit, 3000f, groundLayer))

{

distanceToGround = hit.distance;

if (distanceToGround == 0)

{

isOnGround = true;

}

else

{

isOnGround = false;

}

}

}

void HandleEngine()

{

if (Input.GetKeyDown(KeyCode.X))

{

StartEngine();

} else if(Input.GetKeyDown(KeyCode.Y) /\*&& isOnGround\*/)

{

StopEngine();

}

}

void HelicopterHover()

{

float upForce = 1f - Mathf.Clamp(helicopterRigid.transform.position.y / effectiveHeight,0,1);

upForce = Mathf.Lerp(0,EnginePower, upForce) \* helicopterRigid.mass;

helicopterRigid.AddRelativeForce(Vector3.up \* upForce);

}

void HelicopterMovements()

{

if (Input.GetAxis("Vertical") > 0)

{

helicopterRigid.AddRelativeForce(Vector3.forward \* Mathf.Max(0f, Movement.y \* ForwardForce \* helicopterRigid.mass));

}

else if (Input.GetAxis("Vertical") < 0)

{

helicopterRigid.AddRelativeForce(Vector3.back \* Mathf.Max(0f, -Movement.y \* BackwardForce \* helicopterRigid.mass));

}

float turn = TurnForce \* Mathf.Lerp(Movement.x, Movement.x \* (TurnForceHelper - Mathf.Abs(Movement.y)), Mathf.Max(0f, Movement.y));

turning = Mathf.Lerp(turning, turn, Time.fixedDeltaTime \* TurnForce);

helicopterRigid.AddRelativeTorque(0f, turning \* helicopterRigid.mass, 0f);

}

void HelicopterTilting()

{

TILTING.y = Mathf.Lerp(TILTING.y, Movement.y \* ForwardTiltForce, Time.deltaTime);

TILTING.x = Mathf.Lerp(TILTING.x, Movement.x \* TurnTiltForce, Time.deltaTime);

helicopterRigid.transform.localRotation = Quaternion.Euler(TILTING.y, helicopterRigid.transform.localEulerAngles.y, -TILTING.x);

}

public void StartEngine ()

{

DOTween.To(Starting, 0, 8.0f, engineStartSpeed);

}

void Starting (float value)

{

EnginePower = value;

}

public void StopEngine()

{

DOTween.To(Stopping, EnginePower, 0.0f, engineStartSpeed);

}

void Stopping(float value)

{

EnginePower = value;

}

}

1. VoiceMovement.cs

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.Linq;

using UnityEngine.Windows.Speech;

using UnityEngine.Animations;

public class VoiceMovement : MonoBehaviour

{

public Vector3 thirdViewPosition = new Vector3(10f, 20f, 50f);

public Vector3 firstViewPosition = new Vector3(0f,0f,0f);

private KeywordRecognizer keywordRecognizer;

private Dictionary<string, Action> actions = new Dictionary<string, Action>();

void Start()

{

actions.Add("switch", SwitchView);

keywordRecognizer = new KeywordRecognizer(actions.Keys.ToArray());

keywordRecognizer.OnPhraseRecognized += RecognizedSpeed;

keywordRecognizer.Start();

}

private void RecognizedSpeed(PhraseRecognizedEventArgs speech)

{

Debug.Log(speech.text);

actions[speech.text].Invoke();

}

private void SwitchView()

{

if (transform.localPosition == thirdViewPosition)

{

transform.localPosition = firstViewPosition;

} else if (transform.localPosition == firstViewPosition)

{

transform.localPosition = thirdViewPosition;

}

}

}

1. FlyingBallon.cs

using UnityEngine;

public class FlyingBalloon : MonoBehaviour

{

public float speed = 5f;

public float detectionRadius = 10f;

public float changeDirectionInterval = 5f;

public float rotationSpeed = 5f;

private Vector3 targetDirection;

private float timeSinceLastDirectionChange;

private void Start()

{

targetDirection = RandomDirection();

timeSinceLastDirectionChange = 0f;

}

private void Update()

{

// Change direction if the time since the last change is greater than the change direction interval

timeSinceLastDirectionChange += Time.deltaTime;

if (timeSinceLastDirectionChange >= changeDirectionInterval)

{

targetDirection = RandomDirection();

timeSinceLastDirectionChange = 0f;

}

// Move the balloon in the target direction

transform.position += targetDirection \* speed \* Time.deltaTime;

// Rotate the balloon to face the target direction

Quaternion targetRotation = Quaternion.LookRotation(targetDirection, Vector3.up);

transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, rotationSpeed \* Time.deltaTime);

}

private Vector3 RandomDirection()

{

return new Vector3(Random.Range(-1f, 1f), Random.Range(-1f, 1f), Random.Range(-1f, 1f)).normalized;

}

}

1. Tải bối cảnh từ OSM, cấu hình và làm chi tiết bối cảnh, thiết lập quy tắc vật lý ở bối cảnh chính.
2. Thêm đối tượng phụ, quỹ đạo ngẫu nhiên di chuyển trong bối cảnh.

Đối tượng balloon thêm kịch bản di chuyển ngẫu nhiên và tránh vật thể.