## 领英蓝色系商务宣传画册封面设计

# R

# 342大冒险

# 1.0

制作小组：18342小组

目录：

1. 关于《342大冒险》的介绍
   1. 关于《342大冒险》----------------------------------------------------p.3
   2. 《342大冒险的功能》--------------------------------------------------p.4
2. 我们的unity---------------------------------------------------------------p.5

2.1实机图例演示-------------------------------------------------------------p.6

1. 源文件代码及其解析

3.1人物控制------------------------------------------------------------------p.7

3.2基础相机移动-------------------------------------------------------------p.12

3.3障碍关卡设计-------------------------------------------------------------p.20

3.4碰撞判定------------------------------------------------------------------p.21

1. 我们的专业团队------------------------------------------------------------p.22

## 关于《342大冒险》：

你说的对，但是《342大冒险》是由342寝室自主研发的一款全新跑酷冒险游戏。游戏发生在一个被称作「342」的幻想世界，在这里，被神选中的人将被授予「跑酷之力」，导引肌肉之力不断前进。你将扮演一位名为「张文豪」的神秘角色，在自由的旅行中跑来跑去，不断闯关，找回失散的亲人——同时，逐步发掘「Running」的真相。

## 《342大冒险》的功能：

该项目旨在让平时好吃懒做的你在一个虚拟的世界中实现奔跑的理想

在大冒险中，你可以跑，你可以跳，你甚至可以不断前进！你可以成为自己！实现自己的理想。在你跑完了一个循环后，到达终点时却会再一次回到起点，令人感叹。

别想太多，使劲奔跑吧！

## **我们的开发引擎：**

## 我们使用了unity3d作为我们的开发引擎，Unity是实时3D互动内容创作和运营平台。包括游戏开发、美术、建筑、汽车设计、影视在内的所有创作者，借助Unity将创意变成现实。Unity平台提供一整套完善的软件解决方案 ，可用于创作、运营和变现任何实时互动的[2D](https://baike.baidu.com/item/2D/11445?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/Unity/_blank)和3D内容，[支持平台](https://baike.baidu.com/item/%E6%94%AF%E6%8C%81%E5%B9%B3%E5%8F%B0/5486854?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/Unity/_blank)包括手机、[平板电脑](https://baike.baidu.com/item/%E5%B9%B3%E6%9D%BF%E7%94%B5%E8%84%91/1348389?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/Unity/_blank)、PC、游戏主机、[增强现实](https://baike.baidu.com/item/%E5%A2%9E%E5%BC%BA%E7%8E%B0%E5%AE%9E/1889025?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/Unity/_blank)和虚拟现实设备。

## 我们的专业团队以增量模型为基础，对用户进行需求性分析，以此进行基础设计，在原有的的设备上进行深层次挖掘开发，以实现对用户的需求高度满足，经过黑箱白箱测试，最终我们的《342大冒险》横空出世。

## 实机图例演示：

## 人物三大模型：

## )]9XH_4ND]OV[5QD4`C)K%0

## 障碍关卡：

## ~L(K$SX1TZC}4RDQ20(%$L4

人物建模：

## FNJ}Q7B%Y$O4O55_$[M5{Z7

## 我们的源文件及代码分为四部分：

1.人物控制：

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CharactorMovement : MonoBehaviour

{

[Header("--移动--")]

[SerializeField]

private float speed = 5f; // 移动速度

[SerializeField]

private float turnSpeed = 0.05f; // 旋转速度

[Header("--车道设定--")]

[SerializeField]

private float distance = 5f; // 距离

[SerializeField]

private int direction = 1; // 方向 (0 = 左, 1 = 中, 2 = 右)

[Header("--跳跃设定--")]

[SerializeField]

private float gravity = 12f; // 重力

[SerializeField]

private float jumpForce = 10f; // 跳起来的推力

[Header("--翻滚设定--")]

[SerializeField]

private float rollingTime = 1f; // 翻转时间

[SerializeField]

private float targetCapsuleHeight = 0.5f; // 碰撞体高度

[SerializeField]

private Vector3 targetCapsuleCenter = new Vector3(0, 3f, 0); // 碰撞体位置

[Header("--地上判定--")]

[SerializeField]

private float groundCheckRadius = 0.25f; // 圈圈

[SerializeField]

private float groundCheckOffset = -0.5f; // 碰撞体偏移量

[SerializeField]

private float groundCheckDistance = 0.4f; // 距离

[SerializeField]

private LayerMask groundMask; // 屏蔽

// 确保角色一直接触到地板

private float verticalVelocity = -0.1f; // 垂直重力

private CharacterController characterController; // 脚本控制

private Animator animator;

private float capsuleHeight; // 碰撞体高度

private Vector3 capsuleCenter; // 碰撞体xyz

private Vector3 groundNormal = Vector3.up; // 新增新的 Vector3

private bool isRolling; // 设定布尔值确认是否翻滚

private float rollElapsedTime; // 翻滚定时器

private void Start()

{

// 抓取组件

characterController = GetComponent<CharacterController>();

// 抓取碰撞体高度

capsuleHeight = characterController.height;

// 抓取碰撞体 xyz

capsuleCenter = characterController.center;

animator = GetComponent<Animator>();

}

private void SetMoveDirection(bool isRight)

{

// ? 是 （? :）的简洁写法，用于根据条件选择不同的值进行运算。

direction += isRight ? 1 : -1; // True = 1 , False = -1

// Clamp 界定范围

direction = Mathf.Clamp(direction, 0, 2); // (0 = 左, 1 = 中, 2 = 右)

}

private void SetLookDirection()

{

// 当移动往左或往右时，旋转角色

Vector3 lookDirection = characterController.velocity; // 角色速率

if (lookDirection == Vector3.zero) return; // 如果为0 , return

lookDirection.y = 0;

// Lerp 滑顺

transform.forward = Vector3.Lerp(transform.forward, lookDirection, turnSpeed);

}

private void MoveTo()

{

// 往前移动 只需抓取Z轴 向前动就可以

Vector3 targetPosition = transform.position.z \* Vector3.forward;

// 往左 或 往右

if (direction == 0) targetPosition += Vector3.left \* distance;

else if(direction == 2) targetPosition += Vector3.right \* distance;

// 计算xyz移动数值

Vector3 movement = Vector3.zero;

movement.x = (targetPosition - transform.position).normalized.x \* speed; // 左右移动

movement.y = verticalVelocity; // 垂直速率

movement.z = speed;

// 移动角色

characterController.Move(movement \* Time.deltaTime);

}

private void Jump()

{

// 如果在地板

if (CheckGrounded())

{

// 按下空格键

if (Input.GetKeyDown(KeyCode.Space)) verticalVelocity = jumpForce;

}

// 设定落下时间

else

{

verticalVelocity -= (gravity \* Time.deltaTime);

}

animator.SetBool("IsGround", CheckGrounded());

}

// 确定角色是否在地上

private bool CheckGrounded()

{

// 寻找开始位置 (射线起点)

Vector3 start = transform.position + Vector3.up \* groundCheckOffset;

// 执行圆形射线 start, radius, direction, hit info, distance, layermask (依照官方公式)

if(Physics.SphereCast(start, groundCheckRadius, Vector3.down, out RaycastHit hit, groundCheckDistance, groundMask))

{

groundNormal = hit.normal;

return true;

}

groundNormal = Vector3.up;

return false;

}

private void Roll()

{

if (Input.GetKeyDown(KeyCode.LeftControl))

{

if (isRolling || !CheckGrounded()) return;

rollElapsedTime = 0f;

isRolling = true;

// 如果正在滚动 缩小碰撞体

characterController.height = targetCapsuleHeight;

characterController.center = targetCapsuleCenter;

}

// 计时秒数

rollElapsedTime += Time.deltaTime;

// 重置所有值 (秒数到，回归正常大小)

if(rollElapsedTime>= rollingTime && isRolling)

{

characterController.height = targetCapsuleHeight;

characterController.center = targetCapsuleCenter;

rollElapsedTime = 0f;

isRolling = false;

}

animator.SetBool("IsRolling", isRolling);

}

private void Update()

{

// 用指定移动按键设定方向

if (Input.GetKeyDown(KeyCode.A)) SetMoveDirection(false);

if (Input.GetKeyDown(KeyCode.D)) SetMoveDirection(true);

MoveTo();

SetLookDirection();

Jump();

Roll();

}

// Debug功能

private void OnDrawGizmosSelected()

{

// 设定 gizmos 颜色

Gizmos.color = Color.red;

if (CheckGrounded()) Gizmos.color = Color.green;

// 寻找 开始/结束 位置的圆形射线

Vector3 start = transform.position + Vector3.up \* groundCheckOffset;

Vector3 end = start + Vector3.down \* groundCheckDistance;

// 画圆形导引线

Gizmos.DrawWireSphere(start, groundCheckRadius);

Gizmos.DrawWireSphere(end, groundCheckRadius);

}

}

1. 基础相机移动

#if ENABLE\_INPUT\_SYSTEM

using UnityEngine.InputSystem;

#endif

using UnityEngine;

public class SimpleCameraController : MonoBehaviour

{

class CameraState

{

public float yaw;

public float pitch;

public float roll;

public float x;

public float y;

public float z;

public void SetFromTransform(Transform t)

{

pitch = t.eulerAngles.x;

yaw = t.eulerAngles.y;

roll = t.eulerAngles.z;

x = t.position.x;

y = t.position.y;

z = t.position.z;

}

public void Translate(Vector3 translation)

{

Vector3 rotatedTranslation = Quaternion.Euler(pitch, yaw, roll) \* translation;

x += rotatedTranslation.x;

y += rotatedTranslation.y;

z += rotatedTranslation.z;

}

public void LerpTowards(CameraState target, float positionLerpPct, float rotationLerpPct)

{

yaw = Mathf.Lerp(yaw, target.yaw, rotationLerpPct);

pitch = Mathf.Lerp(pitch, target.pitch, rotationLerpPct);

roll = Mathf.Lerp(roll, target.roll, rotationLerpPct);

x = Mathf.Lerp(x, target.x, positionLerpPct);

y = Mathf.Lerp(y, target.y, positionLerpPct);

z = Mathf.Lerp(z, target.z, positionLerpPct);

}

public void UpdateTransform(Transform t)

{

t.eulerAngles = new Vector3(pitch, yaw, roll);

t.position = new Vector3(x, y, z);

}

}

const float k\_MouseSensitivityMultiplier = 0.01f;

CameraState m\_TargetCameraState = new CameraState();

CameraState m\_InterpolatingCameraState = new CameraState();

[Header("Movement Settings")]

[Tooltip("Exponential boost factor on translation, controllable by mouse wheel.")]

public float boost = 3.5f;

[Tooltip("Time it takes to interpolate camera position 99% of the way to the target."), Range(0.001f, 1f)]

public float positionLerpTime = 0.2f;

[Header("Rotation Settings")]

[Tooltip("Multiplier for the sensitivity of the rotation.")]

public float mouseSensitivity = 60.0f;

[Tooltip("X = Change in mouse position.\nY = Multiplicative factor for camera rotation.")]

public AnimationCurve mouseSensitivityCurve = new AnimationCurve(new Keyframe(0f, 0.5f, 0f, 5f), new Keyframe(1f, 2.5f, 0f, 0f));

[Tooltip("Time it takes to interpolate camera rotation 99% of the way to the target."), Range(0.001f, 1f)]

public float rotationLerpTime = 0.01f;

[Tooltip("Whether or not to invert our Y axis for mouse input to rotation.")]

public bool invertY = false;

#if ENABLE\_INPUT\_SYSTEM

InputAction movementAction;

InputAction verticalMovementAction;

InputAction lookAction;

InputAction boostFactorAction;

bool mouseRightButtonPressed;

void Start()

{

var map = new InputActionMap("Simple Camera Controller");

lookAction = map.AddAction("look", binding: "<Mouse>/delta");

movementAction = map.AddAction("move", binding: "<Gamepad>/leftStick");

verticalMovementAction = map.AddAction("Vertical Movement");

boostFactorAction = map.AddAction("Boost Factor", binding: "<Mouse>/scroll");

lookAction.AddBinding("<Gamepad>/rightStick").WithProcessor("scaleVector2(x=15, y=15)");

movementAction.AddCompositeBinding("Dpad")

.With("Up", "<Keyboard>/w")

.With("Up", "<Keyboard>/upArrow")

.With("Down", "<Keyboard>/s")

.With("Down", "<Keyboard>/downArrow")

.With("Left", "<Keyboard>/a")

.With("Left", "<Keyboard>/leftArrow")

.With("Right", "<Keyboard>/d")

.With("Right", "<Keyboard>/rightArrow");

verticalMovementAction.AddCompositeBinding("Dpad")

.With("Up", "<Keyboard>/pageUp")

.With("Down", "<Keyboard>/pageDown")

.With("Up", "<Keyboard>/e")

.With("Down", "<Keyboard>/q")

.With("Up", "<Gamepad>/rightshoulder")

.With("Down", "<Gamepad>/leftshoulder");

boostFactorAction.AddBinding("<Gamepad>/Dpad").WithProcessor("scaleVector2(x=1, y=4)");

movementAction.Enable();

lookAction.Enable();

verticalMovementAction.Enable();

boostFactorAction.Enable();

}

#endif

void OnEnable()

{

m\_TargetCameraState.SetFromTransform(transform);

m\_InterpolatingCameraState.SetFromTransform(transform);

}

Vector3 GetInputTranslationDirection()

{

Vector3 direction = Vector3.zero;

#if ENABLE\_INPUT\_SYSTEM

var moveDelta = movementAction.ReadValue<Vector2>();

direction.x = moveDelta.x;

direction.z = moveDelta.y;

direction.y = verticalMovementAction.ReadValue<Vector2>().y;

#else

if (Input.GetKey(KeyCode.W))

{

direction += Vector3.forward;

}

if (Input.GetKey(KeyCode.S))

{

direction += Vector3.back;

}

if (Input.GetKey(KeyCode.A))

{

direction += Vector3.left;

}

if (Input.GetKey(KeyCode.D))

{

direction += Vector3.right;

}

if (Input.GetKey(KeyCode.Q))

{

direction += Vector3.down;

}

if (Input.GetKey(KeyCode.E))

{

direction += Vector3.up;

}

#endif

return direction;

}

void Update()

{

// Exit Sample

if (IsEscapePressed())

{

Application.Quit();

#if UNITY\_EDITOR

UnityEditor.EditorApplication.isPlaying = false;

#endif

}

// Hide and lock cursor when right mouse button pressed

if (IsRightMouseButtonDown())

{

Cursor.lockState = CursorLockMode.Locked;

}

// Unlock and show cursor when right mouse button released

if (IsRightMouseButtonUp())

{

Cursor.visible = true;

Cursor.lockState = CursorLockMode.None;

}

// Rotation

if (IsCameraRotationAllowed())

{

var mouseMovement = GetInputLookRotation() \* k\_MouseSensitivityMultiplier \* mouseSensitivity;

if (invertY)

mouseMovement.y = -mouseMovement.y;

var mouseSensitivityFactor = mouseSensitivityCurve.Evaluate(mouseMovement.magnitude);

m\_TargetCameraState.yaw += mouseMovement.x \* mouseSensitivityFactor;

m\_TargetCameraState.pitch += mouseMovement.y \* mouseSensitivityFactor;

}

// Translation

var translation = GetInputTranslationDirection() \* Time.deltaTime;

// Speed up movement when shift key held

if (IsBoostPressed())

{

translation \*= 10.0f;

}

// Modify movement by a boost factor (defined in Inspector and modified in play mode through the mouse scroll wheel)

boost += GetBoostFactor();

translation \*= Mathf.Pow(2.0f, boost);

m\_TargetCameraState.Translate(translation);

// Framerate-independent interpolation

// Calculate the lerp amount, such that we get 99% of the way to our target in the specified time

var positionLerpPct = 1f - Mathf.Exp((Mathf.Log(1f - 0.99f) / positionLerpTime) \* Time.deltaTime);

var rotationLerpPct = 1f - Mathf.Exp((Mathf.Log(1f - 0.99f) / rotationLerpTime) \* Time.deltaTime);

m\_InterpolatingCameraState.LerpTowards(m\_TargetCameraState, positionLerpPct, rotationLerpPct);

m\_InterpolatingCameraState.UpdateTransform(transform);

}

float GetBoostFactor()

{

#if ENABLE\_INPUT\_SYSTEM

return boostFactorAction.ReadValue<Vector2>().y \* 0.01f;

#else

return Input.mouseScrollDelta.y \* 0.01f;

#endif

}

Vector2 GetInputLookRotation()

{

// try to compensate the diff between the two input systems by multiplying with empirical values

#if ENABLE\_INPUT\_SYSTEM

var delta = lookAction.ReadValue<Vector2>();

delta \*= 0.5f; // Account for scaling applied directly in Windows code by old input system.

delta \*= 0.1f; // Account for sensitivity setting on old Mouse X and Y axes.

return delta;

#else

return new Vector2(Input.GetAxis("Mouse X"), Input.GetAxis("Mouse Y"));

#endif

}

bool IsBoostPressed()

{

#if ENABLE\_INPUT\_SYSTEM

bool boost = Keyboard.current != null ? Keyboard.current.leftShiftKey.isPressed : false;

boost |= Gamepad.current != null ? Gamepad.current.xButton.isPressed : false;

return boost;

#else

return Input.GetKey(KeyCode.LeftShift);

#endif

}

bool IsEscapePressed()

{

#if ENABLE\_INPUT\_SYSTEM

return Keyboard.current != null ? Keyboard.current.escapeKey.isPressed : false;

#else

return Input.GetKey(KeyCode.Escape);

#endif

}

bool IsCameraRotationAllowed()

{

#if ENABLE\_INPUT\_SYSTEM

bool canRotate = Mouse.current != null ? Mouse.current.rightButton.isPressed : false;

canRotate |= Gamepad.current != null ? Gamepad.current.rightStick.ReadValue().magnitude > 0 : false;

return canRotate;

#else

return Input.GetMouseButton(1);

#endif

}

bool IsRightMouseButtonDown()

{

#if ENABLE\_INPUT\_SYSTEM

return Mouse.current != null ? Mouse.current.rightButton.isPressed : false;

#else

return Input.GetMouseButtonDown(1);

#endif

}

bool IsRightMouseButtonUp()

{

#if ENABLE\_INPUT\_SYSTEM

return Mouse.current != null ? !Mouse.current.rightButton.isPressed : false;

#else

return Input.GetMouseButtonUp(1);

#endif

}

}

1. 障碍关卡设计

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Spawner : MonoBehaviour

{

[Header("关卡"), SerializeField]

private GameObject[] levels;

[Header("关卡停留时间"), SerializeField]

private float stayTime = 15f;

// 是否已经生成关卡

private bool isSpawn;

// 经过的时间

private float elapsedTime;

public void SpawnLevel()

{

if (levels == null) return;

int randomIndex = Random.Range(0, levels.Length);

GameObject level = Instantiate(levels[randomIndex], transform.position, transform.rotation) as GameObject;

// 移除生成对象时名称赋予 "Clone"

level.name = level.name.Replace("(Clone)", "").Trim();

isSpawn = true;

}

private void Update()

{

if (!isSpawn) return;

// 摧毁关卡

elapsedTime += Time.deltaTime;

print("elapsed time: " + elapsedTime);

// 如果 elapsedTime 大于等于 设定好的时间

if (elapsedTime >= stayTime)

{

elapsedTime = 0;

isSpawn = false;

Destroy(transform.parent.parent.gameObject);

}

}

}

1. 碰撞判定

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Events;

using UnityEngine.SceneManagement;

public class TriggerVolume : MonoBehaviour

{

public UnityEvent<GameObject> OnEnter;

private void OnTriggerEnter(Collider other)

{

if(other.TryGetComponent(out CharactorMovement player))

{

OnEnter.Invoke(player.gameObject);

}

}

public void LoadScene(string sceneName)

{

SceneManager.LoadScene(sceneName);

}

}

## 我们的专业团队：

沈恺：队长兼任技术开发----------------------100分

孙宁 ：技术顾问及PPT制作-------------------100分

利松林：技术顾问----------------------------100分

裘逸川：任务汇报汇总------------------------100分

张文豪：汇报文档编写------------------------100分