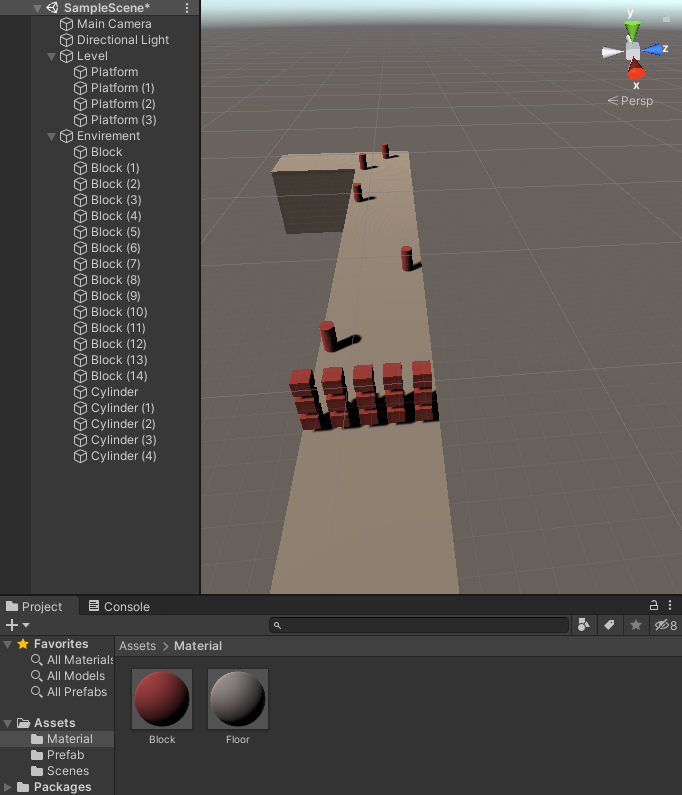
**1. Предисловие**

Сложный проект. Ассеты в материалах, в описаниях. Пример игры:



**2. Создаем Сцену**

Реализуем небольшой уровень для нашей игры, для этого используем стандартные объекты движка Unity.

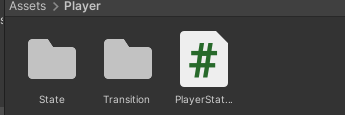


**3. Создаем персонажа**

В материалах имеется готовая модель персонажа, к которому мы добавляем компонент BoxCollider и настирываем его.

**4. Как работает машина состояний**

Добавляем необходимые папки и скрипты, для реализации машины состояний.



Объясняется принцип работы с машиной состояний…

**5. Пишем переходы**

Для реализации добавляем следующие скрипты и код:

|  |
| --- |
| PlayerState.cs |
| public class PlayerState : MonoBehaviour  {  } |

|  |
| --- |
| PlayerTransition.cs |
| public abstract class PlayerTransition: MonoBehaviour {  [SerializeField] private PlayerState targetState;  public PlayerState TargetState => targetState;  public bool NeedTranst { get; protected set; }  private void OnEnable() {  NeedTranst = false;  Enable();  }  public abstract void Enable();  } |
| Особенности Кода:  -класс является абстрактным параметрами, которые можно только считать и редактировать только дочерние классы |

**6. Пишем состояния**

|  |
| --- |
| PlayerState.cs |
| public abstract class PlayerState : MonoBehaviour {  [SerializeField] private PlayerTransition[] transition;  protected Rigidbody Rigidbody { get; private set; }  protected Animator Animator { get; private set; }  public void Enter(Rigidbody rigidbody, Animator animator) {  if (enabled == false) {  Rigidbody = rigidbody;  Animator = animator;  enabled = true;  foreach (var transition in transition) {  transition.enabled = true;  }  }  }  public void Exit() {  if (enabled == true) {  foreach (var transition in transition) {  transition.enabled = false;  }  enabled = false;  }  }  public PlayerState GetNextState() {  foreach (var transition in transition) {  if (transition.NeedTranst) {  return transition.TargetState;  }  }  return null;  }  } |
| Особенности кода:  - |

**7. Дописываем машину состояний**

|  |
| --- |
| PlayerStateMachine.cs |
| [RequireComponent(typeof(Rigidbody), typeof(Animator))]  public class PlayerStateMachine : MonoBehaviour{  [SerializeField] private PlayerState firstState;  private PlayerState currentState;  private Rigidbody rigidbody;  private Animator animator;  public void Awake() {  rigidbody = GetComponent<Rigidbody>();  animator = GetComponent<Animator>();  }  public void Start() {  currentState = firstState;  currentState.Enter(rigidbody, animator);  }  private void Update() {  if (currentState != null) {  return;  }  PlayerState nextState = currentState.GetNextState();  if (nextState != null) {  Transit(nextState);  }  }  private void Transit(PlayerState nextState) {  if (currentState != null) {  currentState.Exit();  }  currentState = nextState;  if (currentState != null) {  currentState.Enter(rigidbody, animator);  }  }  } |
| Особенности кода:  - |

**8. Состояние Idle и Move**

|  |
| --- |
| IdleState |
| public class IdleState : PlayerState {  } |

|  |
| --- |
| MoveState.cs |
| public class MoveState : PlayerState {  [SerializeField] private PlayerInput playerInput;  [SerializeField] private float maxSppeed;  [SerializeField] private float speedRatio;  private void OnEnable() {  playerInput.DirectionChange += OnDirectionChanges;  }  private void OnDisable(){  playerInput.DirectionChange -= OnDirectionChanges;  }  private void OnDirectionChanges(Vector2 direction) {  Rigidbody.velocity = new Vector3(direction.x, 0, direction.y) \* speedRatio;  if (Rigidbody.velocity.magnitude > maxSppeed) {  Rigidbody.velocity \*= maxSppeed / Rigidbody.velocity.magnitude;  }  if (Rigidbody.velocity.magnitude != 0) {  Rigidbody.MoveRotation(Quaternion.LookRotation(Rigidbody.velocity, Vector3.up));  }  }  } |
| Особенности кода:  - |
| PlayerInput.cs |
| public class PlayerInput : MonoBehaviour{  private Vector3 tapPosition;  public event UnityAction<Vector2> DirectionChange;  public event UnityAction PointerUp;  private void Update() {  if (Input.GetMouseButtonDown(0)) {  tapPosition = Input.mousePosition;  }  if (Input.GetMouseButton(0)) {  DirectionChange?.Invoke(Input.mousePosition - tapPosition);  }  if (Input.GetMouseButtonUp(0)){  PointerUp?.Invoke();  }  }  } |
| Особенности кода:  - |

**9. Переход к бегу, донастройка машины состояний**

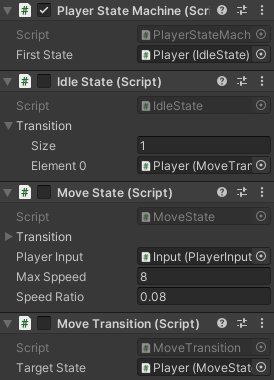
Реализуем добавляя и изменяя следующие скрипты:

|  |
| --- |
| MoveTransition.cs |
| public class MoveTransition : PlayerTransition {  public override void Enable(){  }  private void Update(){  if (Input.GetMouseButtonDown(0)) {  NeedTranst = true;  }  }  } |
| Особенности кода:  - |

|  |
| --- |
| IdleState.cs |
| public class IdleState : PlayerState {  private void Update()  {  }  } |
| Особенности кода:  - |

|  |
| --- |
| PlayerStateMachine.cs |
| ///  private void Update() {  if (currentState == null) {  return;  }  /// |

При этом меняем компоненты в Unity следующим образом:



**10. Накапливаем энергию и переходим к атаке**

Добавляем скрипты и реализуем:

|  |
| --- |
| StaminaAccumulator.cs |
| public class StaminaAccumulator : MonoBehaviour {  [SerializeField] private float accumulationTime;  private float staminaValue;  public void StartAccumulat() {  staminaValue = 0;  }  private void Update() {  staminaValue += Time.deltaTime;  }  public void GetAbbility() {  if (staminaValue > accumulationTime) {  Degug.Log(“Ultimate);  }  Degug.Log(“Simple”);  }  } |
| Особенности кода:  - |

|  |
| --- |
| MoveState.cs |
| ///  [SerializeField] private StaminaAccumulator staminaAccumulator;  private void OnEnable() {  playerInput.DirectionChange += OnDirectionChanges;  staminaAccumulator.StartAccumulat();  }  /// |
| Особенности кода:  - |

|  |
| --- |
| AttackTransition.cs |
| public class AttackTransition : PlayerTransition {  public override void Enable() {  }  private void Update() {  if (Input.GetMouseButtonUp(0)) {  NeedTranst = true;  }  }  } |
| Особенности кода:  - |

|  |
| --- |
| AttackState.cs |
| public class AttackState: PlayerState{  [SerializeField] private StaminaAccumulator staminaAccumulator;  } |
| Особенности кода:  - |

Добавляем необходимые компоненты к нашему Игроку.

**11. Создаем SO атаки и донастриваем аккумулятор**

Дополняем наши скрипты и видоизменяем

|  |
| --- |
| Abillity.cs |
| public class AttackState : PlayerState {  [SerializeField] private StaminaAccumulator staminaAccumulator;  private Abillity currentAbility;  public event UnityAction CollisionDetected;  public event UnityAction AbilityEnded;  private void OnEnable(){  currentAbility = staminaAccumulator.GetAbbility();  }  } |
| Особенности кода:  - ScriptableObject (ссылка на данный файл, для подключение к нему) |

|  |
| --- |
| StaminaAccumulator.cs |
| ///  [SerializeField] private Abillity ability;  [SerializeField] private Abillity ultimateAbility;  public Abillity GetAbbility() {  if (staminaValue > accumulationTime) {  return ultimateAbility;}  return ability;  }  /// |
| Особенности кода:  - |

**12. Hand Ability**

|  |
| --- |
| HandAbility.cs |
| public class HandAbility : Abillity {  [SerializeField] private float attackForce;  [SerializeField] private float UsefulTime;  private AttackState state;  private Coroutine coroutine;  public override event UnityAction AbiliotyEnded;  public override void UseAbility(AttackState attack) {  if (coroutine != null) {  Reset();  }  state = attack;  state.CollisionDetected += OnPlayerAttack;  }  private void OnPlayerAttack() {    }  private IEnumerator Attack(AttackState state) {  float time = UsefulTime;  while (time > 0) {  state.Rigidbody.velocity = state.Rigidbody.velocity.normalized \* attackForce;  time -= Time.deltaTime;  yield return new WaitForEndOfFrame();  }  Reset();  AbiliotyEnded?.Invoke();  }  private void Reset() {  state.Rigidbody.velocity = Vector3.zero;  state.StopCoroutine(coroutine);  coroutine = null;  state.CollisionDetected -= OnPlayerAttack;  }  } |
| Особенности кода:  - |

|  |
| --- |
| PlayerState.cs |
| public abstract class PlayerState : MonoBehaviour {  ///  public Rigidbody Rigidbody { get; private set; }  /// |
|  |

**13. Атакуем всё, что движется, а что не движется движем и атакуем**

Добавляем скрипты и редактируем:

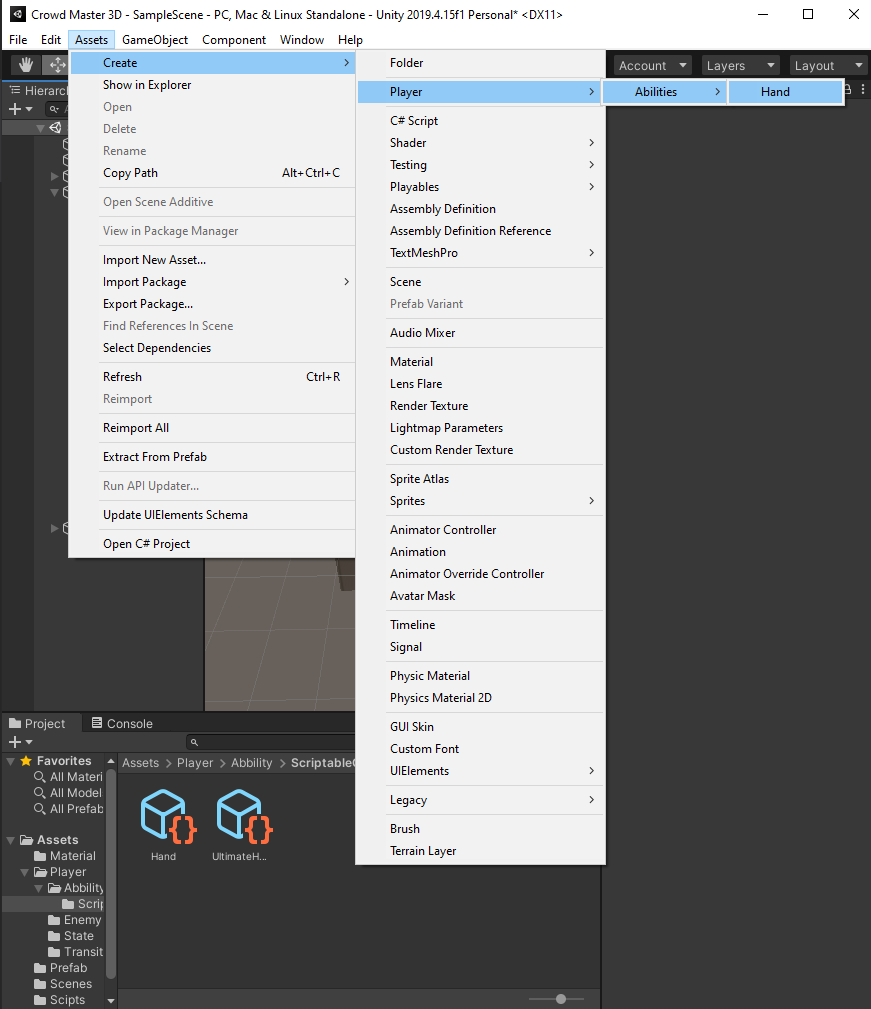
|  |
| --- |
| IDamageable.cs |
| public interface IDamageable {  bool ApplyyDamage(Rigidbody rigidbody, float force);  } |
| Особенности кода:  - |

|  |
| --- |
| Box.cs |
| public class Box : MonoBehaviour, IDamageable {  public bool ApplyyDamage(Rigidbody rigidbody, float force) {  Debug.Log("Corobka");  return true;  }  } |
| Особенности кода:  - |

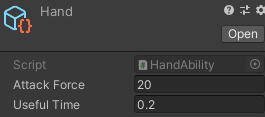
|  |
| --- |
| AttackState.cs |
| public class AttackState : PlayerState {  [SerializeField] private StaminaAccumulator staminaAccumulator;  private Abillity currentAbility;  public event UnityAction<IDamageable> CollisionDetected;  public event UnityAction AbilityEnded;  private void OnEnable() {  currentAbility = staminaAccumulator.GetAbbility();  currentAbility.AbiliotyEnded += OnAbbilityEnded;  currentAbility.UseAbility(this);  }  private void OnDisable() {  currentAbility.AbiliotyEnded -= OnAbbilityEnded;  }  private void OnAbbilityEnded() {  AbilityEnded?.Invoke();  }  private void OnTriggerEnter(Collider other) {  if (other.gameObject.TryGetComponent(out IDamageable damagble))  CollisionDetected?.Invoke(damagble);  }  private void OnCollisionEnter(Collision collision) {  if (collision.gameObject.TryGetComponent(out IDamageable damagble))  CollisionDetected?.Invoke(damagble);  }  private void Update() {    }  } |
| Особенности кода:  - |

|  |
| --- |
| HandAbility.cs |
| [CreateAssetMenu(fileName = "New Hand Ability", menuName = "Player/Abilities/Hand", order = 51)]  public class HandAbility : Abillity {  ///  public override void UseAbility(AttackState attack) {  if (coroutine != null) {  Reset();  }  state = attack;  coroutine = state.StartCoroutine(Attack(state));  state.CollisionDetected += OnPlayerAttack;  }  private void OnPlayerAttack(IDamageable damagble) {  if (damagble.ApplyyDamage(state.Rigidbody, attackForce) == false) {  return;  }  state.Rigidbody.velocity /= 2;  }  /// |
| Особенности кода:  - |

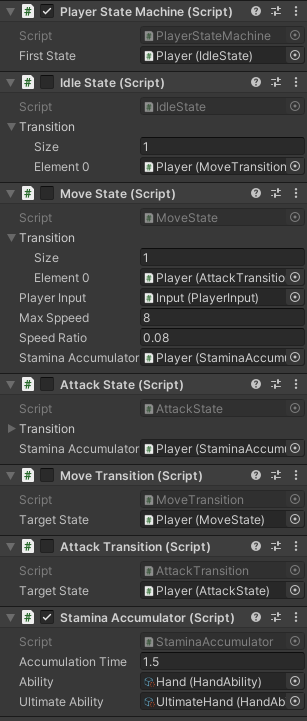
Настройка Компонентов в Unity. Т.к. теперь HandAbility есть некий компонент, то его можно вызвать через основную вкладку, чтобы создать ScriptableObject.



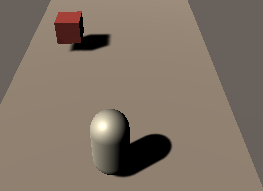
Настраиваем данные компоненты



Настраиваем нашего персонажа:



И создаем нейтральный куб со скриптом Box.

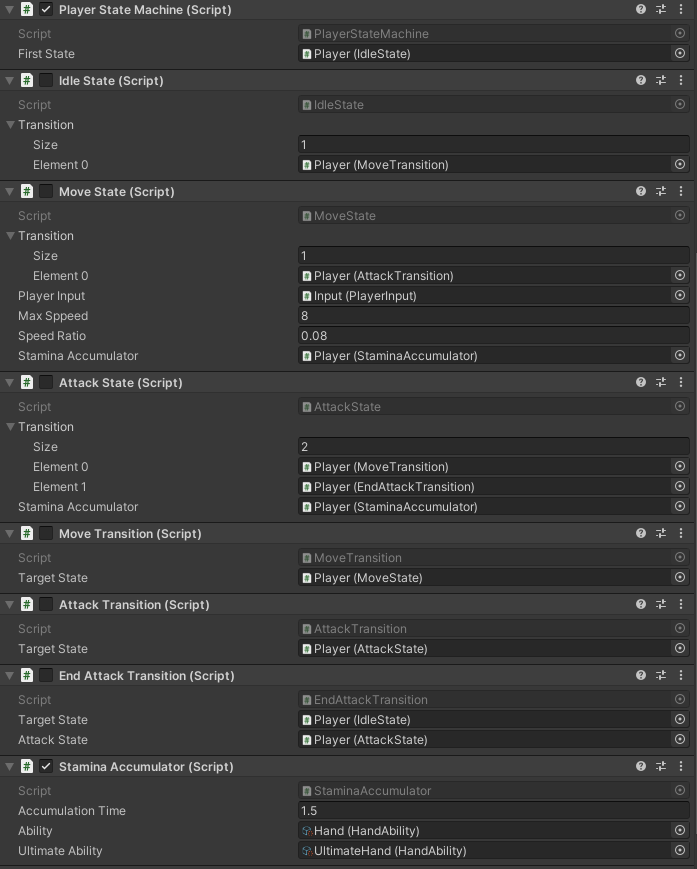


**14. Зацикливаем машину состояний**

Реализуем дополнительный скрипт и донастриваем компоненты нашего персонажа:

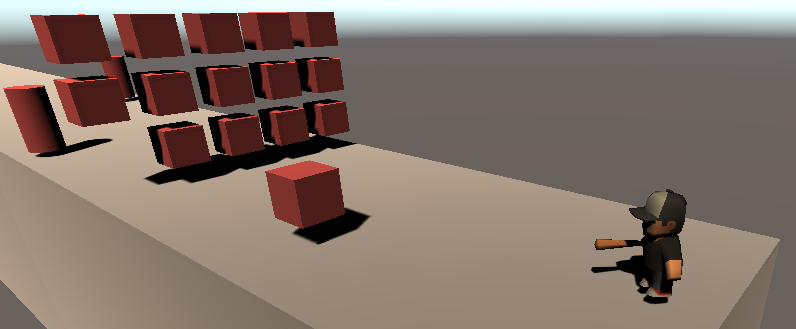
|  |
| --- |
| EndAttackTransition.cs |
| public class EndAttackTransition : PlayerTransition{  [SerializeField] private AttackState attackState;  public override void Enable() {  attackState.AbilityEnded += OnAbbilityEnded;  }  private void OnDisable() {  attackState.AbilityEnded -= OnAbbilityEnded;  }  private void OnAbbilityEnded() {  NeedTranst = true;  }  void Update() {    }  } |
| Особенности кода:  - |

Реализация компонента:



Перед тем, как начать реализовывать анимацию, скачай ради примера Ассет «Toony Tiny People Demo».

И переопределим все необходимые компоненты.



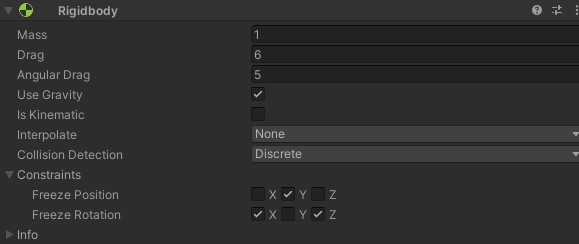
**15. Анимация бега и удара**

В материалах данного курса заранее подготовлены материалы по контроллеру анимации и т.д.

Связи с тем, реализовывать анимацию мы не будем.

**16. Избавление от вращения**

Выставляем следующие параметры нашему Rigidbody персонажа

****

**17. Enemy State Machine**

Добавляем нового персонажа и разделяем классы, для структуризации.

Реализуем…

///////////////////////////////////////////////////------------------------------------------------/////////////////////////

**18. Быдло получает по заслугам**

Реализуем состояния и переходы…

///////////////////////////////////////////////////------------------------------------------------/////////////////////////

**19. Преследование игрока**

Реализуем состояние и переходы… При этом переходы через NavMeshAgent.

///////////////////////////////////////////////////------------------------------------------------/////////////////////////

**20. Атакуем игрока**

Реализуем состояние и переходы…

///////////////////////////////////////////////////------------------------------------------------/////////////////////////

**21. Смерть игрока**

Добавляем нашему персонажа скрипт HealtContainer.cs и редактируем следующий скрипт:

|  |
| --- |
| PlayerStateMachine.cs |
| ///  private HealthContainer health;  private void OnEnable() {  health.Deid += OnDied;  }  private void OnDisable() {  health.Deid -= OnDied;  }    private void OnDied() {  enabled = false;  //animator.SetTrigger("broken");  }  public void Awake() {  ///  health = GetComponent<HealthContainer>();  }  ///  public void ApplyDamage(float damage) {  health.TakeDamage((int)damage);  }  /// |

**22. Камера трясется и следит за героем**

Добавляем два скрипта для Тряски и Слежения

|  |
| --- |
| CameraSheker.cs |
| public class CameraSheker : MonoBehaviour{  [SerializeField] private PlayerStateMachine Player;  [SerializeField] private float Speed;  [SerializeField] private int Range;  private Coroutine coroutine;  private void OnEnable() {  Player.Damaged += OnPlayerDamage;  }  private void OnDisable() {  Player.Damaged -= OnPlayerDamage;  }  private void OnPlayerDamage() {  if (coroutine != null) {  StopCoroutine(coroutine);  }  coroutine = StartCoroutine(Shake());  }  private IEnumerator Shake() {  int shakeRange = Range;  Quaternion targetRotation;  while (shakeRange != 0){  targetRotation = Quaternion.Euler(transform.rotation.eulerAngles.x, transform.rotation.eulerAngles.y, shakeRange);  transform.rotation = Quaternion.RotateTowards(transform.rotation, targetRotation, Speed \* Time.deltaTime);  if (transform.rotation == targetRotation) {  shakeRange = (Mathf.Abs(shakeRange) - 1) \* -1;  }  yield return new WaitForEndOfFrame();  }  transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles.x, transform.eulerAngles.y, 0);  coroutine = null;  }  } |

|  |
| --- |
| CameraFolowing.cs |
| public class CameraFolowing: MonoBehaviour{  [SerializeField] private Rigidbody PLAYER;  [SerializeField] private Vector3 forwardDirection;  [SerializeField] private float speed;  [SerializeField] private float angle;  [SerializeField] private float distance;  [SerializeField] private float maxVecotLength = 2;  private Vector3 nextPosition;  private void Start() {  float rotationY = Mathf.Rad2Deg \* Mathf.Asin(forwardDirection.x / forwardDirection.magnitude);  transform.rotation = Quaternion.Euler(angle, rotationY, transform.rotation.eulerAngles.z);  }  private void FixedUpdate() {  nextPosition = PLAYER.position + Vector3.ClampMagnitude(PLAYER.velocity, maxVecotLength);  nextPosition += Vector3.up \* Mathf.Cos(Mathf.Deg2Rad \* angle) \* distance;  nextPosition += -forwardDirection \* Mathf.Sin(Mathf.Deg2Rad \* angle) \* distance;  transform.position = Vector3.Lerp(transform.position, nextPosition, speed \* Time.fixedDeltaTime);  }  } |

**23. Украшаем жизнь**

Используем и реализуем воду из Aссета MobileDepthWater.

Используем сайт color.cu и украшаем материалы наших препятствий.

Используем Gradient и реализуем освещение.