**1. Демонстрация оригинала**



**2. Создание блоков**

Добавляем необходимые структурные компоненты в игру:

1. Поле, где будет стоять башня
2. Башня, состоящая из одного куба
3. Блок-препятствие

Структурируем папки:

1. Scripts + Block.cs + Spawner.cs
2. Prefab (Башня и блок препятствия)
3. Materials для Башни и Блока-препятствия

**3. Создание спаунера**

Для создания Спаунера редактируем код Spawner.cs

|  |
| --- |
| Spawner.cs |
| public class Spawner : MonoBehaviour {  [SerializeField] private Block SegmentTempate;  [SerializeField] private Block Block;  [SerializeField] private Block FinishTemplate;  [SerializeField] private int TowerSize;  private void Start() {  BuildTower();  }  private void BuildTower() {  GameObject currentpoint = gameObject;  for (int i = 0; i < TowerSize; i++) {  currentpoint = BuildSegment(currentpoint, SegmentTempate.gameObject);  currentpoint = BuildSegment(currentpoint, Block.gameObject);  }  BuildSegment(currentpoint, FinishTemplate.gameObject);  }  private GameObject BuildSegment(GameObject currentSegment, GameObject nextSegment) {  return Instantiate(nextSegment, GetBuildPoint(currentSegment.transform, nextSegment.transform), Quaternion.identity, transform);  }  private Vector3 GetBuildPoint(Transform currentSegment, Transform nextSegment) {  return new Vector3(transform.position.x, currentSegment.position.y + currentSegment.localScale.y / 2 + nextSegment.localScale.y / 2, transform.position.z);  }  } |

Далее соединяем скрипт с нашим объектом Spawner и заполняем переменные объектами.

Во время выполнения редактируем материалы, позиции и названия объектов, для упорядоченности.

**4. Полет мяча**

Реализации мяча будет сделано через Подкидывание мяча через GetMouseButtonDown().

Основная задача это остановиться при следующем задаче, для реализации данной задачи используем переход ФизиченскогоТела из Dynamic в Kinematic.

Реализуем данную программу в скрипте:

|  |
| --- |
| Ball.cs |
| [RequireComponent(typeof(Rigidbody))]  public class Ball: MonoBehaviour{  private Rigidbody rigidbody;  [SerializeField] private float JumpForce;  private void Start() {  rigidbody = GetComponent<Rigidbody>();  }  private void Update() {  if (Input.GetMouseButtonDown(0)) {  rigidbody.isKinematic = false;  rigidbody.AddForce(Vector2.up \* JumpForce, ForceMode.Impulse);  }  if (Input.GetMouseButtonUp(0)) {  Ray ray = new Ray(transform.position, Vector3.forward);  if (Physics.Raycast(ray, out RaycastHit hitInfo)) {  if (hitInfo.collider.TryGetComponent(out Block block)) {  rigidbody.isKinematic = true;  rigidbody.velocity = Vector3.zero;  }    }    }  }  } |

Добавляем скрипт на наш мяч и редактируем RayCast, чтобы он показывал в правильную сторону.

Пробуем.

**5. Разделяем башню на компоненты**

Для этого добавляем новые скрипты Finish.cs и Segment.cs. Добавляем их к нашим Префабам и редактируем код в скриптакх, где они используются, а в скрипте Ball.cs реализуем попадание по различным объектам:

|  |
| --- |
| Ball.cs |
| if (hitInfo.collider.TryGetComponent(out Block block)){  ///< Реализация  Debug.Log("GameOver");  }  else if (hitInfo.collider.TryGetComponent(out Segment segment)) {  rigidbody.isKinematic = true;  rigidbody.velocity = Vector3.zero;  } else if (hitInfo.collider.TryGetComponent(out Finish fifnish)) {  ///< Реализация  Debug.Log("You Win");  } |

**6. Добавляем шест**

В материалах урока дается готовый объект, имеющий анимацию.

Для реализации механики мы используем простой цилиндр, который будут добавлен в качестве дочернего объекта для объекта Ball и в зависимости от положения нашего объекта Ball будет включать или выключать наш объект.

|  |
| --- |
| Ball.cs |
| [RequireComponent(typeof(Rigidbody))]  public class Ball: MonoBehaviour{  private Rigidbody rigidbody;  [SerializeField] private float JumpForce;  [SerializeField] private GameObject Tree;  private void Start() {  rigidbody = GetComponent<Rigidbody>();  TreeActiveFalse();  }  private void Update() {  if (Input.GetMouseButtonDown(0)) {  TreeActiveFalse();  rigidbody.isKinematic = false;  rigidbody.AddForce(Vector2.up \* JumpForce, ForceMode.Impulse);  }  if (Input.GetMouseButtonUp(0)) {  Ray ray = new Ray(transform.position, Vector3.forward);  if (Physics.Raycast(ray, out RaycastHit hitInfo)) {  if (hitInfo.collider.TryGetComponent(out Segment segment)) {  rigidbody.isKinematic = true;  rigidbody.velocity = Vector3.zero;  Tree.SetActive(true);  }  else if (hitInfo.collider.TryGetComponent(out Block block)) {  Debug.Log("GameOver");  TreeActiveFalse();  } else if (hitInfo.collider.TryGetComponent(out Finish fifnish)) {  ///< Реализация  Debug.Log("You Win");  TreeActiveFalse();  }  }  }  }  private void TreeActiveFalse() {  Tree.SetActive(false);  }  } |

В случаи с готовыми материалами реализуется следующим образом:

1. К объекту добавляется комопнент Аниматор
2. Создаем новую папку Stack и создаем внутри него Аниматор Контроллер с названием Стик и связываем палку с нашим контроллером.
3. Открываем Аниматор
4. Для реализации добавляем новую состояние Empty и назначаем анимацию из нашего объекта.
5. Для реализации «сгибания» от значения сделаем следующее.
6. Создаем FromNewBuildTree и переходим внутрь него.
7. У него есть параметр Blend и добавляем AddMotion и добавляем нашу анимацию.
8. При этом BlandType мы выбираем Direct.
9. Привязываем объект к нашему мячу (Внимательно! Последняя кость и ест верхушка).

Дополнительная информация:

1. Как получить переменную с нашего аниматора

|  |
| --- |
| Exampl.cs |
| [SerializeField] private Animator animator;  [SerializeField] private float power;  private void Update() {  animator.SetFloat("Blend",power);  // Math.Clamp(); ///< Он принимает value, min, max  } |

**7. Отслеживания шарика камерой**

Пишем скрипт для нашего BallTracker.cs

|  |
| --- |
| BallTracker.cs |
| public class BallTracker : MonoBehaviour {  [SerializeField] private Ball ball;  [SerializeField] private float speed;  private void FixedUpdate() {  Vector3 targetPosition = new Vector3(transform.position.x,ball.transform.position.y+3, transform.position.z);  transform.position = Vector3.Lerp(transform.position, targetPosition, speed \* Time.fixedDeltaTime);  }  } |

Настраиваем и доделываем проект с добавлением необходимых компонентов.