Сделал все так что при присоединении игрока, ему сетался рандомный материал. Рандомное число индекса материала задается на сервере. При присоединении игрока на сервер игрок передает длину массива и в этом диапозоне рандомится индекс материала

```
export class Player extends Schema {
    @type("number") x = Math.floor(Math.random() * 256) -128;
    @type("number") z = Math.floor(Math.random() * 256) -128;
    @type("uint8") d = 2;
    @type("uint8") clr = 0;
}

export class State extends Schema {
    @type({ map: Player })
    players = new MapSchema<Player>();

    something = "This attribute won't be sent to the client-side";    Alexander [8 days

    createPlayer(sessionId: string, data: any) {
        this.players.set(sessionId, new Player());
        this.players.get(sessionId).clr = Math.floor(Math.random() * data.skin);
    }
```

Этот же момент у клиента

```
private async void Connection() {

    Dictionary<string, object> data = new Dictionary<string, object>(){
        { "skin", Skins.Instance.GetLength() }
    };
    _room = await Instance.client.JoinOrCreate<State>(GameRoomName, data);
    _room.OnStateChange += OnChange;
}
```

Далее в классе MultiplayerManager при создании змейки игрока и енеми мы передаем это число:

```
Snake snake = Instantiate(_snakePrefab, position, rotation);
snake.Init(player.d, player.clr);
```

И затем при создании всех частей змейки (Snake, Detail, Tail) передаю это число:

```
public void Init(int detailCount, byte color) {
    GetComponent<SetSkin>().SetMaterial(color);

    _tail = Instantiate(_tailPrefab, transform.position, Quaternion.identity);
    _tail.Init(_head, _speed, detailCount, color);
}
```

Есть 2 класса: Skins и SetSkin Класс Skins просто весит на сцене, хранит возможные материалы, возвращает необходимый и возвращает длину массива для генерации рандомного индекса на сервере, вынес его отдельно, чтобы добавлять материалы можно было в одном месте

```
1 asset usage  3 usages  Alexander *  1 exposing API
public class Skins : MonoBehaviour {
    public static Skins Instance;

    [SerializeField] private Material[] _materials;  Serializable

    Event function  Alexander
    private void Awake() {
        if (Instance == null) {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else {
            Destroy(gameObject);
        }
    }

    1 usage  new *
    public byte GetLength() => (byte) _materials.Length;

    1 usage  new *
    public Material GetMaterial(byte index) => _materials[index];

}
```

А класс SetSkin весит на этих префабах, хранит необходимые меши, на которые устанавливает выбранный материал

```
public class SetSkin : MonoBehaviour {
    [SerializeField] private MeshRenderer [] _meshRenderers;  Serializable

    1 usage  Alexander
    public void Set(Material material){
        for (int i = 0; i < _meshRenderers.Length; i++){
            _meshRenderers[i].material = material;
        }
    }
    3 usages  Alexander
    public void SetMaterial(byte index) => Set(Skins.Instance.GetMaterial(index));
}
```