Для начала я изменил немного класс MapInfo и добавил в него два метода по добавлению и удалению зданий и юнитов из списка, что позволяет избежать ошибок при их уничтожении

```
    [field: SerializeField] public List<Tower> _enemyTowers { get; private set; } = new List<Tower>();
    [field: SerializeField] public List<Tower> _playerTowers { get; private set; } = new List<Tower>();

    [field: SerializeField] public List<Unit> _enemyUnits { get; private set; } = new List<Unit>();
    [field: SerializeField] public List<Unit> _playerUnits { get; private set; } = new List<Unit>();

    public void AddToList<T>(List<T> list, T obj) => list.Add(obj);
    public void RemoveFromList<T>(List<T> list, T obj) => list.Remove(obj);
```

Так же очень удобно, что при создании юнита он будет на старте добавляться в этот список; Вот реализация в классе Unit, где на старте добавляем этого юнита в соответствующий список:

```
    private UnitState _currentState;
    private MapInfo _info;

    private void Start() {
        _info = MapInfo.Instance;
        if(!_isEnemy) _info.AddToList(_info._playerUnits, obj: this);
        else _info.AddToList(_info._enemyUnits, obj: this);

        defaultState = Instantiate( defaultStateSO);
```

А в методе OnDestroy удаляем его из этого списка:

```
    private void OnDestroy() {
        if(!_isEnemy) _info.RemoveFromList(_info._playerUnits, obj: this);
        else _info.RemoveFromList(_info._enemyUnits, obj: this);
    }
```

По аналогии сделал с классом Tower:

```
[RequireComponent(typeof(Health))]
public class Tower : MonoBehaviour, IHealth {
    [field: SerializeField] public bool _isEnemyTower { get; private set; } = false;
    [field: SerializeField] public Health Health { get; private set; }

    [field: SerializeField] public float Radius { get; private set; } = 2f;

    public float GetDistance(in Vector3 point) => Vector3.Distance(a: transform.position, b: point) - Radius;

    private MapInfo _info;
    private void Start() {
        _info = MapInfo.Instance;
        if(!_isEnemyTower) _info.AddToList(_info._playerTowers, obj: this);
        else _info.AddToList(_info._enemyTowers, obj: this);
    }

    private void OnDestroy() {
        if(!_isEnemyTower) _info.RemoveFromList(_info._playerTowers, obj: this);
        else _info.RemoveFromList(_info._enemyTowers, obj: this);
    }
}
```

Так теперь про HP
Юниты и здания хранят ссылку на префаб HealthBar и на старте создают его в выбранной позиции и припэренчивают к самому себе и чтобы игрок всегда видел HealthBar – его угол порота равен углу поворота камеры

```
public class LookAtCamera : MonoBehaviour {
    private Transform _camera;
    private void Start() => _camera = Camera.main.transform;
    private void Update() => transform.rotation = _camera.rotation;
}
```

Сама логика HealthBar сделана как в шутере только не с Canvas, а с 3D object – Quad, а так все тоже самое, вот скрипт

```
public class HealthBar : MonoBehaviour {
    [SerializeField] private Transform _foreground;

    public void UpdateHealth(float current, float max){
        float percent = current / max;
        _foreground.localScale = new Vector3( x: Mathf.Clamp01(percent), _foreground.localScale.y, _foreground.localScale.z);
    }
}
```

Ну и в классе Health как по мне самое идеальное место для создания этого префаба и возможности обновлять HealthBar при получении дамага; так же тут идет проверка на 0 и если 0 или ниже то удаляется юнит или здание, вот скрипт:

```
public class Health : MonoBehaviour {
    [field: SerializeField] public float Max { get; private set; } = 10f;
    [SerializeField] private HealthUI _healthUIPrefab;
    [SerializeField] private Transform _healthUIPosition;
    private HealthUI _healthUI;
    private float _current;

    private void Start() {
        _current = Max;
        if(_healthUIPrefab)
            _healthUI = Instantiate(_healthUIPrefab, _healthUIPosition.position, Quaternion.identity, transform);
    }

    public void ApplyDamage(float value) {
        _current -= value;

        if (_current < 0) {
            _current = 0;
            Destroy(gameObject);
        }

        Debug.Log( message: $"Объект {name}:было {_current + value} , стало {_current}");
        if(_healthUI)
            _healthUI.UpdateHealth(_current, Max);
    }

}
```