1. **Create a pool interface that will be used for coin entities**

```
public interface IEntityPool
{
   public IEntity Rent();
   public void Return(IEntity entity);
}
```

2. **Create a pool implementation that will be used for coin entities**

```
using System.Collections.Generic;
using Atomic.Entities;
using UnityEngine;

namespace GameExample.Engine
{
   public sealed class SceneEntityPool : IEntityPool
   {
       private readonly SceneEntity prefab;

       private readonly Transform worldContainer;
       private readonly Transform poolContainer;

       private readonly Queue<SceneEntity> queue = new();

       public SceneEntityPool(
           SceneEntity prefab,
           Transform poolContainer,
           Transform worldContainer,
           int initalCount = 0
       )
       {
           this.prefab = prefab;
           this.poolContainer = poolContainer;
           this.worldContainer = worldContainer;

           for (int i = 0; i < initialCount; i++)
           {
               SceneEntity entity = SceneEntity.Instantiate(this.prefab,
this.poolContainer);
               this.queue.Enqueue(entity);
           }
       }

       public IEntity Rent()
       {
           if (this.queue.TryDequeue(out SceneEntity entity))
           {
               entity.transform.SetParent(this.worldContainer);
               return entity;
           }

           return SceneEntity.Instantiate(this.prefab, this.worldContainer);
       }

       public void Return(IEntity entity)
       {
           SceneEntity sceneEntity = SceneEntity.Cast(entity);
           sceneEntity.transform.SetParent(this.poolContainer);
```

```
            this.queue.Enqueue(sceneEntity);
        }
    }
}
```

3. Create data for Coin System

```
[Serializable]
public sealed class CoinSystemData
{
    public IEntityPool pool;
    public Bounds spawnArea;
    public Cycle spawnCycle;
}
```

4. Write coin spawn methods

```
public static class SpawnCoinUseCase
{
    public static IEntity SpawnCoinInArea(this IContext gameContext)
    {
        float3 spawnPoint = gameContext.RandomCoinSpawnPoint();
        return gameContext.SpawnCoin(spawnPoint);
    }

    public static IEntity SpawnCoin(this IContext gameContext, float3 spawnPoint)
    {
        IEntityPool coinPool = gameContext.GetCoinSystemData().pool;
        IEntity coin = coinPool.Rent();
        coin.GetPosition().Value = spawnPoint;
        return coin;
    }

    private static float3 RandomCoinSpawnPoint(this IContext gameContext)
    {
        Bounds spawnArea = gameContext.GetCoinSystemData().spawnArea;
        float3 min = spawnArea.min;
        float3 max = spawnArea.max;
        float3 spawnPoint = new float3(Random.Range(min.x, max.x), 0,
Random.Range(min.z, max.z));
        return spawnPoint;
    }
}
```

5. Create a system that spawns a coin entity at random place every X time

```
public sealed class CoinSpawnSystem : IContextInit, IContextEnable,
IContextDisable, IContextUpdate
{
    private IContext _gameContext;
    private Cycle _spawnPeriod;

    public void Init(IContext context)
    {
        _gameContext = context;
        _spawnPeriod = context.GetCoinSystemData().spawnCycle;
    }

    public void Enable(IContext context)
    {
        _spawnPeriod.Start();
        _spawnPeriod.OnCycle += this.Spawn;
    }

    public void Update(IContext context, float deltaTime)
    {
        _spawnPeriod.Tick(deltaTime);
    }

    public void Disable(IContext context)
    {
        _spawnPeriod.Stop();
        _spawnPeriod.OnCycle -= this.Spawn;
    }

    private void Spawn()
    {
        _gameContext.SpawnCoinInArea();
    }
}
```

Create installer that adds coin system state and behavior to GameContext

```
[Serializable]
public sealed class CoinSystemInstaller : IContextInstaller
{
    [SerializeField]
    private SceneEntity coinPrefab;

    [SerializeField]
    private int initialPoolCount;

    [SerializeField]
    private Transform poolTransform;

    [SerializeField]
    private float spawnPeriod = 2;

    [SerializeField]
    private Bounds spawnArea = new(Vector3.zero, new Vector3(5, 0, 5));
```

```
    public void Install(IContext context)
    {
        var worldTransform = context.GetWorldTransform();
        var coinData = new CoinSystemData
        {
            pool = new SceneEntityPool(this.coinPrefab, this.poolTransform,
worldTransform, this.initialPoolCount),
            spawnArea = this.spawnArea,
            spawnCycle = new Cycle(this.spawnPeriod)
        };

        context.AddCoinSystemData(coinData);
        context.AddSystem<CoinSpawnSystem>();
    }
}
```