Марчук Д. К., ст. викладач,
Лантвойт К. М., здобувач
Державний університет «Житомирська політехніка»

РОЛЬ DICONTAINERS В ІГРОВІЙ ІНДУСТРІЇ ТА КОРИСТЬ ЇХ ВИКОРИСТАННЯ В АРХІТЕКТУРІ ПРОЕКТУ

Архітектура коду та побудова правильних залежностей в коді є одним із найважливіших етапів розробки гри. Залежності класів (компонентів) повинні бути зрозумілими та легко ініціалізованими, щоб інші компоненти мали легкий доступ до компонентів (об'єктів) і чітку структуру з одним місцем ін'єкції залежностей та зв'язків між об'єктами. Рішенням може бути DI Containers (Dependency Injection Container) – контейнер для ін'єкції залежностей, який спрощує та автоматизує впровадження залежностей в об'єкти програми за допомогою деякого контейнера або системи керування контейнерами [1]. На даний момент DI Containers широко використовується в різних сферах розробки програмного забезпечення починаючи з веброзробки і закінчуючи розробкою ігор, тощо.

Проблема розглянута в даному дослідженні – яку роль відіграє DI Containers в ігровій індустрії, коли використовується та є корисним.

DI Containers допомагають надати єдине місце входу ін'єкцій в програмі. Одним із фреймворків що реалізують DI Containers є Zenject, створений для рушія Unity, він дозволяє ефективно управляти залежностями в грі та несе велику функціональну базу. Zenject має велику кількість функцій, методі та класів, що є невеликим мінусом але в той самий момент і плюсом даного фреймворку, так як розширяє його можливостей до великих масштабів. Найбільш часто використовуваними та важливими функціями в є:

- Ін'єкція залежностей надає потужний механізм ін'єкції залежностей, який дозволяє автоматично вставляти залежності в об'єкти під час їх створення
- Контексти для залежностей для визначення які залежності повинні використовуватися певними частинами проекту.
- Керування одиночками (Singeltons) можна виконувати за допомогою Zenject, що робить їх легшими у використані та управлінні.
- Тестування. Робить тестування зручнішим завдяки заміни реальних залежностей на тестові.
- Масштабованість. Надає змогу підтримувати масштабовані проекти та забезпечує гнучку архітектуру проекту.

І це тільки мала частина можливостей Zenject, але в більшості випадків використовують вище вказані функції так як вони є найбільш актуальними та важливими в розробці ігор. DI широко використовується не тільки в малих проектах, а і у великих проектах та компаніях, так як надає зручні функції в сфері тестування та масштабування, що є дуже важливим під час розробки гри, так як хороше тестування є запорукою успіху проекту і є майже фінальним етапом розробки гри, а масштабування надає змогу реалізувати багато ігрових механік і при цьому мати чітку та зрозумілу архітектуру проекту. Для прикладу можна розглянути використання Zenject під час розробки гри.

Створення зв'язку між IInputController та його реалізацією:

```
private void BindDesktop() {
      DesktopInput desktopInput = Container.InstantiatePrefabForComponent<DesktopInput>(
         _desktopInput, _desktopInput.transform.position, Quaternion.identity, _inputContainer);
Container.Bind<IInputController>().To<DesktopInput>().FromInstance(desktopInput).AsSingle();
        }
```

Приклад отримання реалізації:

```
[Inject] public void Construct(
      IInputController inputController) {
      _inputController = inputController;
   }
```

Приклад швидкої заміни:

```
private void BindHandheld() {
      HandheldInput handheldInput = Container.InstantiatePrefabForComponent<HandheldInput>(
         _handheldInput, _handheldInput.transform.position, Quaternion.identity, _inputContainer);
Container.Bind<IInputController>().To<HandheldInput>().FromInstance(handheldInput).AsSingle();
        }
```

Zenject є важливим та часто використовуваним інструментом під час розробки ігор, він надає змогу чітко визначати залежності між об'єктами, створювати зв'язки та надавати реалізації в потрібних місцях коду, допомагає проводити зручне та гнучке тестування проектів, також спрямований на проекти великих масштабів та надає чітку архітектуру за допомогою єдиних місць входу.

Список використаних джерел

1. Arora A. Dependency Injection in Unity 3D [Електронний ресурс] / Akshay Arora. – Режим доступу до ресурсу: https://medium.com/xrpractices/dependency-injection-in-unity-86afabf2cf8d.