Game Overview

Star Connection is a puzzle game where the player must connect stars by rotating nodes along a path. Completing challenges earns the player experience points, unlocks new stages, and allows leveling up when certain experience thresholds are met.

Key Features:

- **Rotate and Connect Nodes**: The objective is to connect stars by rotating path nodes to create a route between two stars.
- **Stage Hub**: Access and select different stages through a central hub.
- **Persistent Progression**: Save player progress and levels, allowing continuity across gaming sessions.
- **Animation and Sound Effects**: Enhanced gaming experience with smooth animations and immersive sound effects.
- **Android Based**: Developed for the Android platform.

Structure overview

External libraries:

- **LeanTween**: Used for efficient tweening animations.

Design patterns used:

- **Observer**: Implemented to ensure game event updates are propagated to interested components.
- **Singleton**: Used to manage classes that require a single global instance, such as the game manager.

Animations

Tweening was used for most of the project's animations, providing smooth transitions.
The Animator was specifically used for the stage completion screen for a more special effect.

Prefabs

Use of Prefabs and Variants: Ensured a consistent production flow and facilitated project maintenance.

Assets

All assets used in this game are either made by me or sourced from free-use libraries.

Class structure

Game management

- **GameManager**: A Singleton class that holds important game data, such as stage list information. Manages macro events in the game, such as loading the next stage in sequence.

Node system

- **Node**: Handles the interaction and rotation logic of the nodes.
- **NodeModel**: A model class used to update the UI and animate the node based on events fired from the main class.
- **NodeManager**: This class manages macro events from the nodes, primarily checking if paths are completed between the edge nodes.
- **NodeConnection and NodeConnectionPoint**: Scripts created to handle connections between nodes, by observing child "NodeConnectionPoint" groups, a “NodeConnection” class manages and handles any connection event.

Stage and progress data

- **StageSO**: Scriptable Object that contains stage information, like display name, scene reference, and experience gained for completing it.
- **ProgressionSO**: Another ScriptableObject that holds the player's persistent data. Once updated, any object subscribed to the class events will update its UI accordingly.

Hub management

- **HubManager**: Responsible for retrieving the stage list from the GameManager and instantiating all of the stage buttons. It uses information from ProgressionSO to update the state of each HubButton, indicating whether it is locked, not completed, or completed.