ROBOLEON:
Technical Design Document

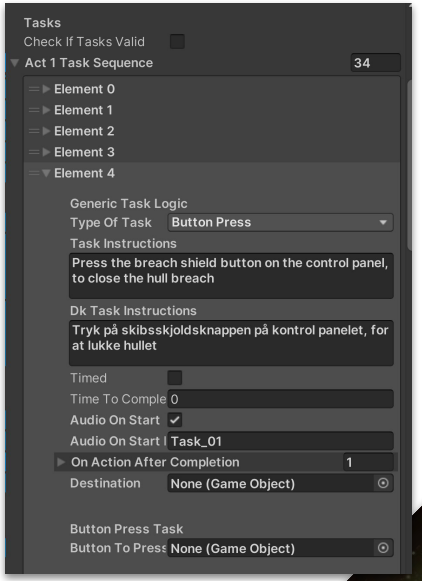
Team 3 Programming

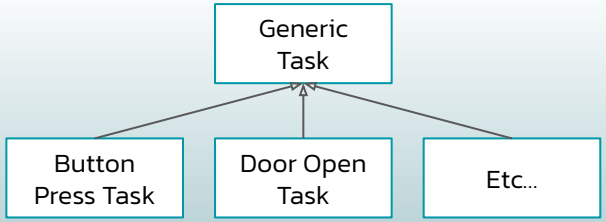
General Overview

- **Modular approach to design**
  - Easy-to-use, scalable utilities for game designers and audio designers
- **Task Manager**
  - Modular task system allows for flexibility and freedom in level design
- **Audio Manager**
  - Swappable audio events facilitate integration with audio team
- **Character Controller**
  - iterations on puppet-feel mechanic
- **Save - Load**
  - scalable saveable data types
- **Procedural Content**
  - spawn fires around spaceship scene
- **UI**
  - responsive to audio and controller feedback and use translations
- **Cinematics**
  - key narrative elements

Technical Components: Modular Task System

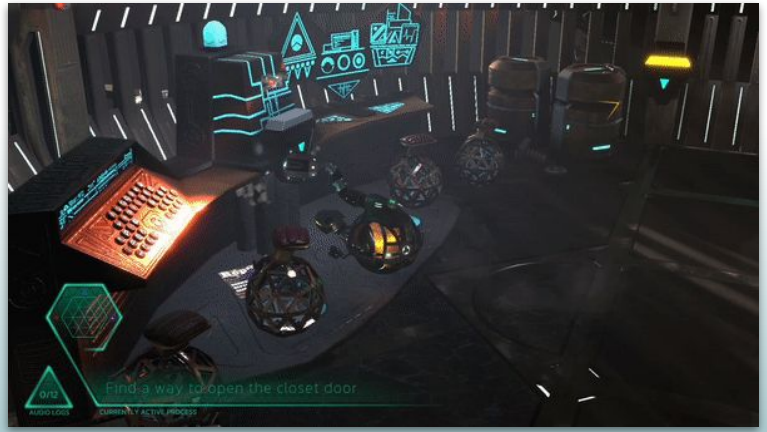
- Task system created using polymorphism
  - 8 task types can be selected from
  - can be added in any order and unlimited in number within the inspector
- Each task can
  - trigger audio events
  - trigger "On Action Events" for interactable objects around the spaceship
- All task types requires specific, unique actions be taken in order to progress





OnAction Scripting

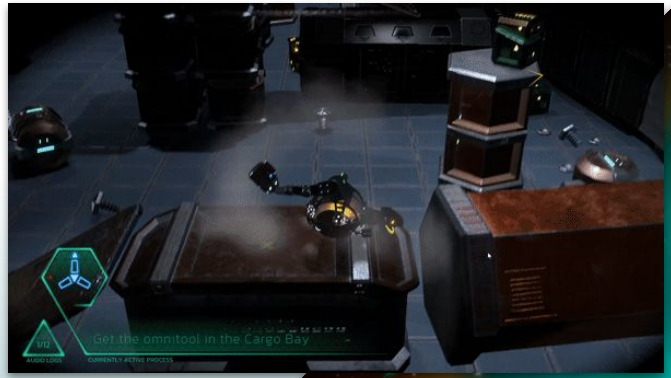
- Objects that need to be activated receive OnAction scripting
  - doors: locked–> unlocked
  - lights: flashing status
  - buttons: not pressed –> pressed
  - game objects: disabled –> enabled (and vice versa)
  - swapping between materials on game objects
- Used to allow the player to interact with the environment
- Shows the environment reacting to the task system



Character Controller

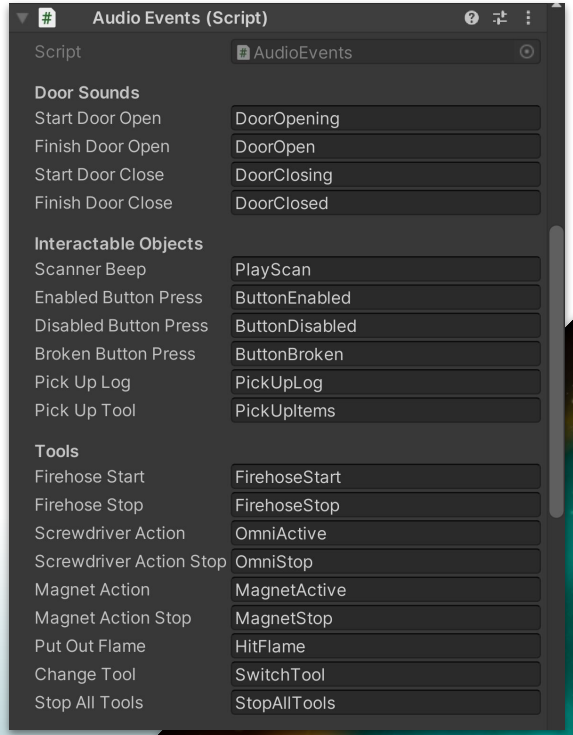
- **100% Physics-based Character**
  - Unpredictable interactions with environment
  - Powered by Unity physics system : *Rigidbody* and *Joint*
  - *Rigidbody* : "Rolling" of character is driven by torque
  - *Joint* : Rotating of tail is following an "invisible" object connected via joint

- **Rich Responses In Different Scenarios**
  - Attachment animation plays when swing the tail
  - Controller rumbles when using attachments
  - Core color of "Rolling Ball" changes when conducting a "sharp turn"
  - Head light to indicate the availability of jumping
  - Sound and spark effect play when colliding with environment
  - Movement is always aligned to camera forward direction – easier to control



Audio Manager

- **One script** is responsible for all interaction with Wwise
- Additional Audio Events script holds **variables for Wwise event** names
    - allows for **independent work paces** for programmers and sound department
    - changes within Wwise can be easily implemented in the code by changing the event name within the inspector, no changes needed in code
    - Simple and easy to add new events
- Messages to Wwise collected in a queue which is triggered each frame by the game manager
    - objects in scene add their sounds to the queue rather than sending them directly to Wwise
    - avoids voice starvation and keeps track of all messages sent to Wwise

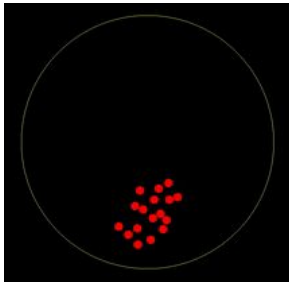


Save and Load

- **Save**
  - Initiated by Game Manager
  - Created a new save if no save exists, or if requested by the main menu
  - Records saved data as a list of saveable objects (i.e. screw panel data) and game data
  - Game data is serialized as JSON in a default persistent data path
  - Autosaves game at customizable time intervals
  - ignores loose/dynamic objects

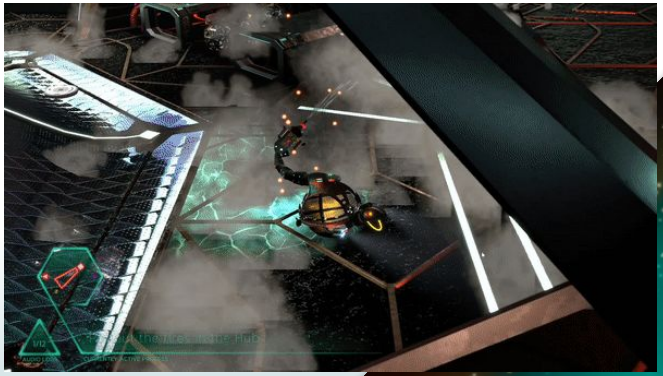
- **Load**
  - Parses JSON from default persistent data path
  - Use task index from game manager save data to simulate task completion
    - simulation of task completion changes scene to previous state

Procedural Content Generation & Fire Spawning



Gameplay mechanics:

- **Poisson Disk Sampling**
  - Pick a point within spawn area
  - Place additional point at minimum distance from current point
  - Remove invalid points
  - Repeat above until spawn area filled
  - Spawn assets at points
- **Fully configurable** by game and level designers
- Fail case if no fire spawn
- Two independent procedural spawners in game:
  - **Fire spawner:** Triggered by a firefighter task in the modular task system
  - **Generic Spawner:** Triggered on scene start, utilized for decoration



Translation Manager

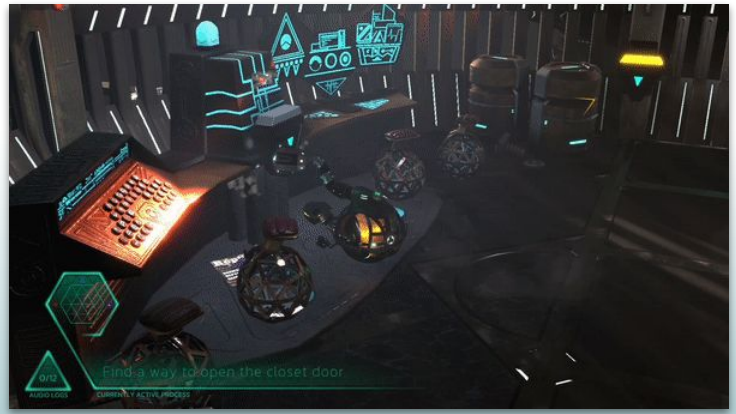
- Keeps track of the current language, updated from the Player Preferences
- Imports JSON files of relevant translations to a dictionary
- UI scripts pass keys to the translation manager to receive the appropriate text for specific elements
- Contains a language change event that UI elements can subscribe to and change themselves when a language change occurs.

Main Menu (3D UI)

- **Canvas** for menu transformed into world space to interact with 3D scene.
- Main Menu script responsible for handling supporting managers (such as the audio manager and translation manager) within the Main Menu scene. These managers are supported by the Game Manager within the level scene
- Additional **helper scripts** are attached to the buttons of the main menu to give the appearance of glowing text on selection to better match the atmosphere of the game.

In-Game UI

- **In-game information**
  - **Objective marker** points to and hovers over designated current task
  - Current task objectives (in text)
  - Current **tail attachment** (symbol)
- **Audio log menu**
  - shows collected + uncollected audio logs
  - replayable audio logs



Cinematics

- Uses playable director objects to control Unity “Timeline” cutscenes
  - in game cutscenes are triggered by collisions with game objects
  - disables player movement during
- Cutscenes are triggered only once