Make-or-Break: A Natural Disaster Simulation Game

Group 5: Esat Duman, Jonathan Juarez, Kent Lizardo, Jose Tejeda

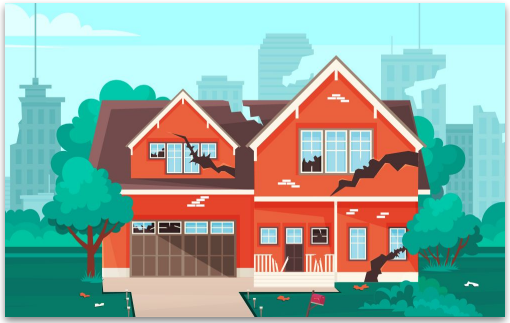
Make or Break: Overview

Make or Break is a game that uses real-time physics simulations in order to test virtual buildings against the forces of nature.

Make or Break allows users to **create, edit, and test** their designs against against natural disasters like earthquakes, floods, or tornadoes.

In **Challenge Mode,** where users can test buildings in order to survive certain challenges of varying difficulties.

In **Sandbox Mode** users get to make buildings with an infinite budget and create disasters of their own.



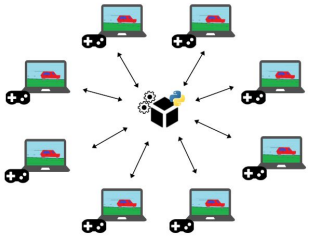
Purpose

Make-or-Break provides a platform that empowers users to simulate and test the structural integrity of buildings against various natural disasters.

By offering a real-time physics simulation, the application aims to assist **designers, engineers, and architects** in designing safer and more resilient structures.

The **goal** behind this project seeks to help create safer environments, while also promoting safety and sustainability in construction practices.

Key Requirements



Precision or Accuracy: Physics calculations must employ fixed physics timestep solutions to ensure accuracy and consistency.

Robustness or Fault-Tolerance: The levels downloaded by the users will still be playable on their systems regardless of server connection.

Adaptability: Make or Break is a PC game so will be compatible with Windows, Mac-OS, and Linux. Will be available on many different gaming platforms like steam, geforce Now, and Epic Games.

Appearance: The appearance should be user friendly, simple and appealing. Fonts throughout the interface should be easy to read. Lighter colors of choice will be used and darker colors will be avoided to make it easy for the client to navigate the game.

Design Goals

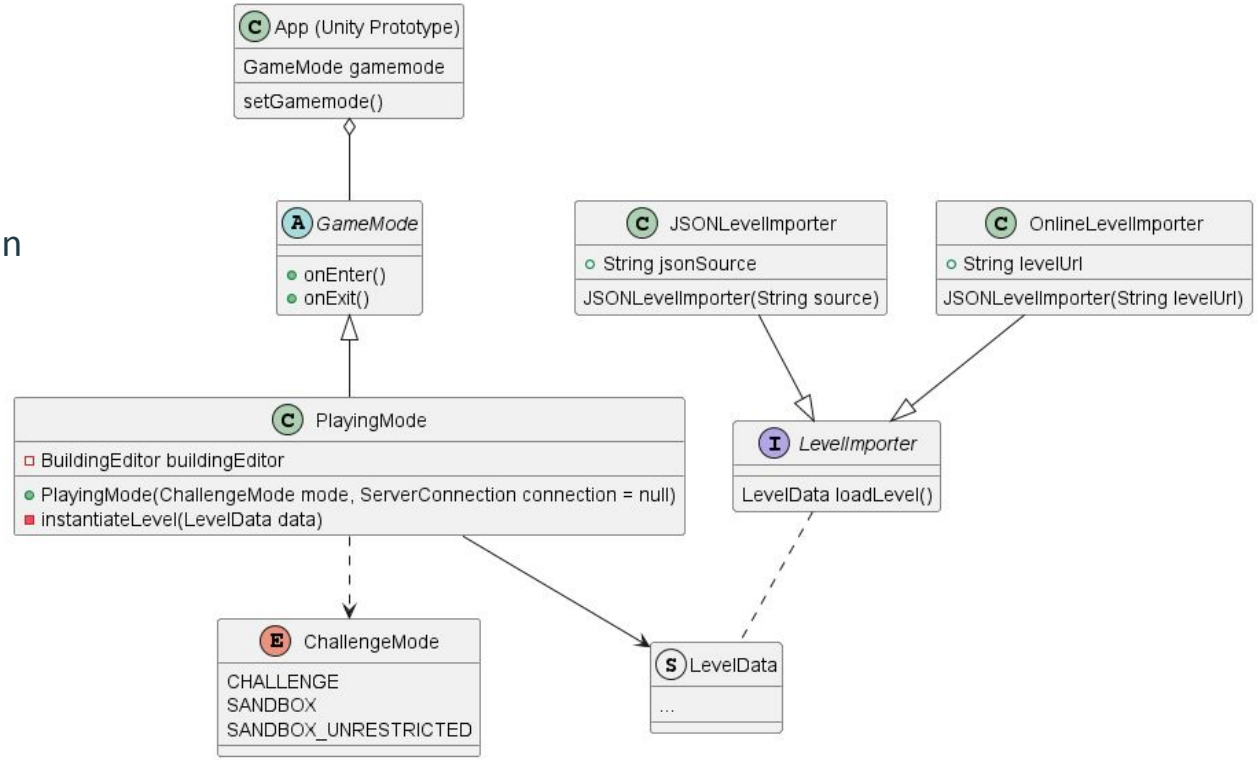
The design of Make-or-Break aims for a balance between **realism and entertainment**. Keeping this in mind the main design objectives for Make or break are:

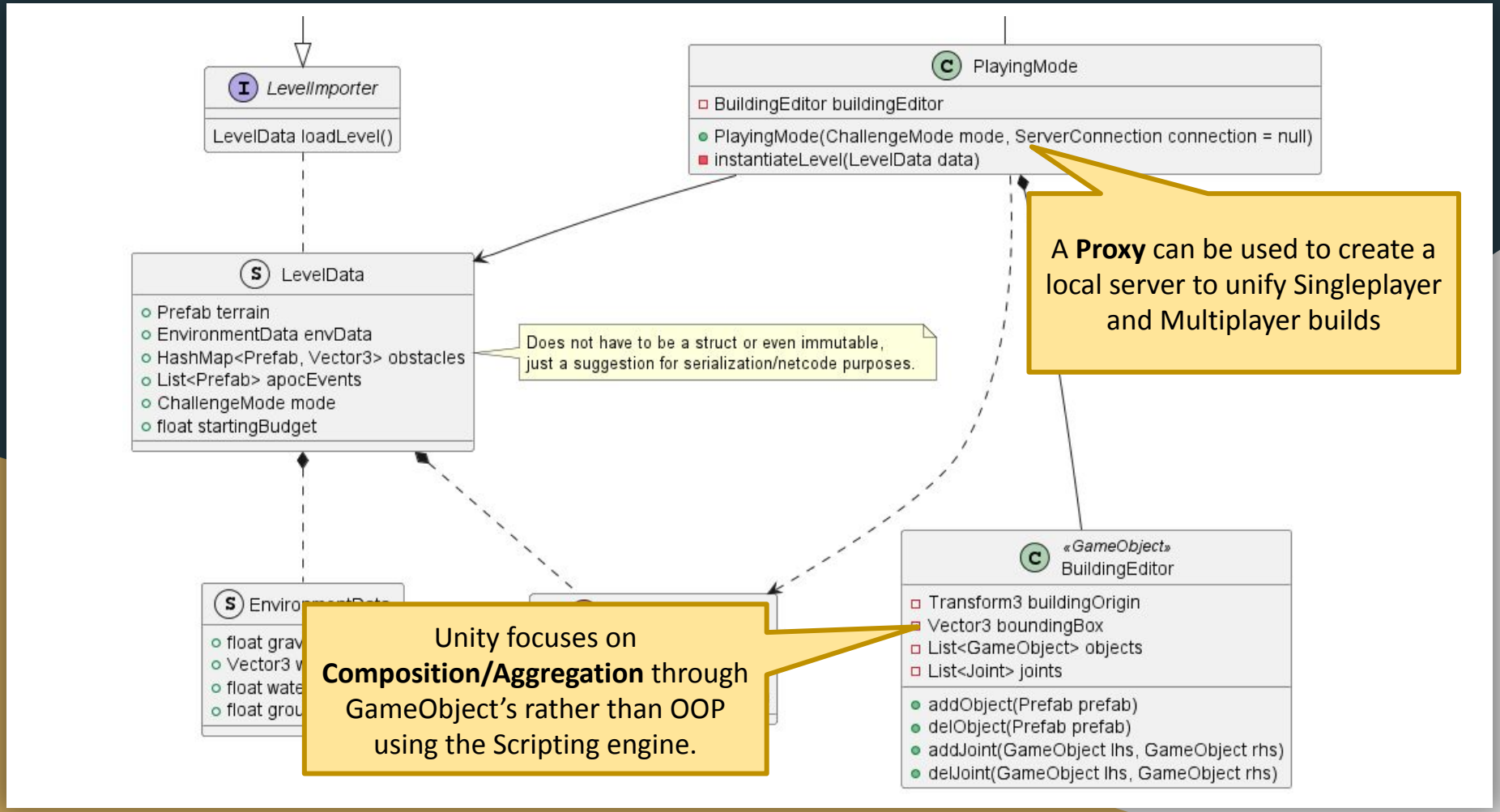
- **Physics Simulations:** Realistic depictions of natural forces on structures
- **Scalability:** Wide range of building sizes / varying magnitudes of disasters
- **Performance Optimization:** Smooth and responsive simulations on varying hardware
- **Education Value:** Introduces engineering principles, and disaster preparedness / awareness

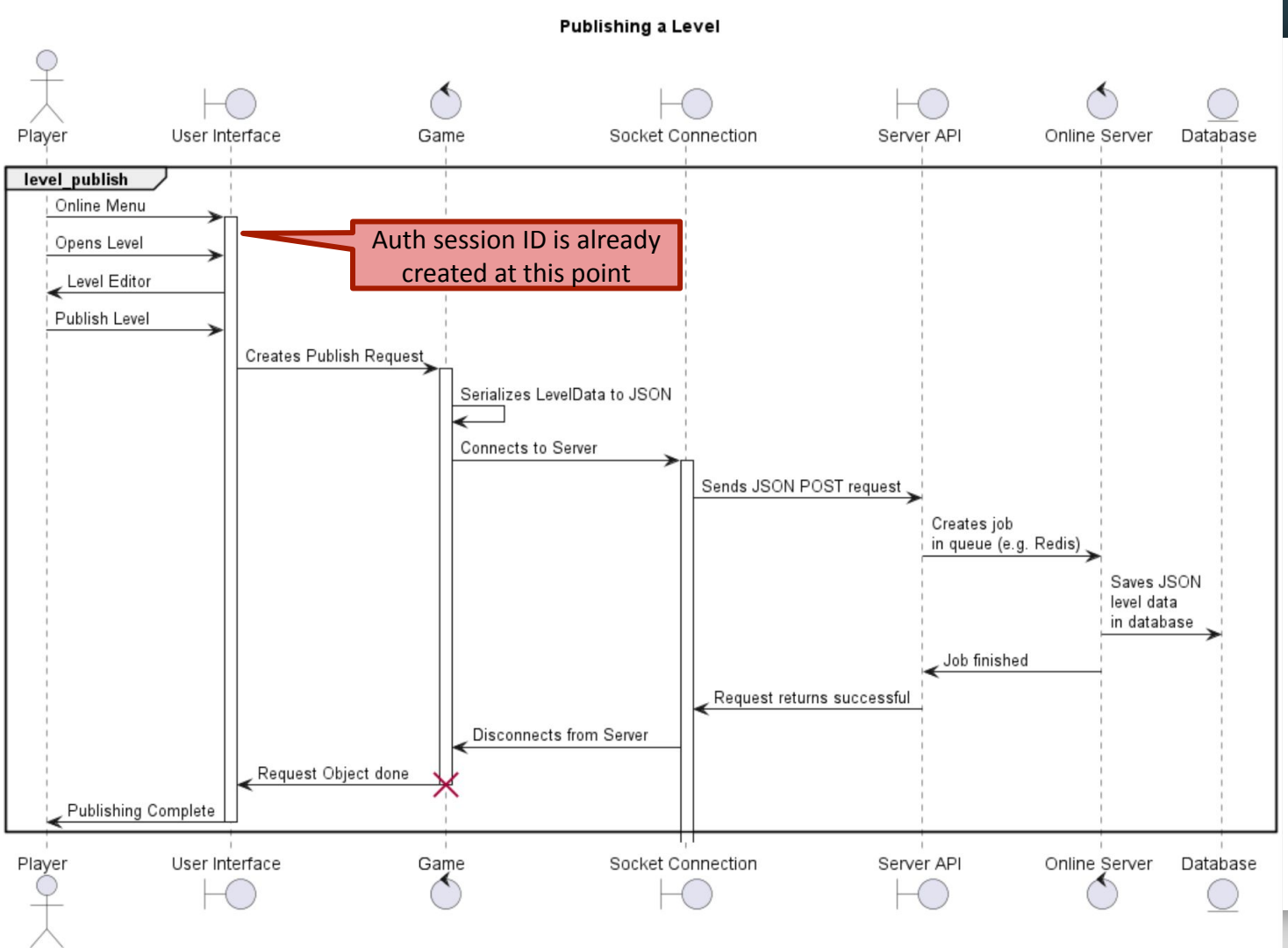
Subsystems - Class Diagram(s)

Possible Concepts/Patterns

- **FSM** for game modes
- **Proxy** for server connection
- **Service Locator** to physics engine calls
- **Aggregation/Composition** for game objects and entities
- **Factory** for importing levels




LevelImporter
LevelData loadLevel()
PlayingMode
BuildingEditor buildingEditor
PlayingMode(ChallengeMode mode, ServerConnection connection = null)
instantiateLevel(LevelData data)
LevelData
Prefab terrain
EnvironmentData envData
HashMap<Prefab, Vector3> obstacles
List<Prefab> apocEvents
ChallengeMode mode
float startingBudget
Does not have to be a struct or even immutable, just a suggestion for serialization/netcode purposes.
A Proxy can be used to create a local server to unify Singleplayer and Multiplayer builds
«GameObject»
BuildingEditor
Transform3 buildingOrigin
Vector3 boundingBox
List<GameObject> objects
List<Joint> joints
addObject(Prefab prefab)
delObject(Prefab prefab)
addJoint(GameObject lhs, GameObject rhs)
delJoint(GameObject lhs, GameObject rhs)
Unity focuses on Composition/Aggregation through GameObject's rather than OOP using the Scripting engine.


Publishing a Level
Player
User Interface
Game
Socket Connection
Server API
Online Server
Database
level_publish
Online Menu
Opens Level
Level Editor
Publish Level
Auth session ID is already created at this point
Creates Publish Request
Serializes LevelData to JSON
Connects to Server
Sends JSON POST request
Creates job in queue (e.g. Redis)
Saves JSON level data in database
Job finished
Request returns successful
Disconnects from Server
Request Object done
Publishing Complete
Player
User Interface
Game
Socket Connection
Server API
Online Server
Database

Potential Issues

Some Factors that could cause **uncertainty or significant issues** in Make or Break's development and success are:

- **Hardware Compatibility:** Consistent performance across varying hardware configurations
- **Resource Constraints:** Limited budgets, time or personnel
- **Technological Dependencies:** Reliance on third party technologies for physics simulations, rendering, multiplayer support, etc.
- **User Feedback:** Affects future updates, features, marketing strategies, etc.

Important Terms

Player: person who buys and plays the game on their system.

Deterministic physics: physics where the outcome is solely determined by the inputs. (no randomness)

Updates/Mods: modifications made to the game post-release, (mods usually by players, and updates/patches by developers)

Real-Time Simulation: Changes made by players are immediately reflected in the game's environment