Six Angry Devs

Obstacle Odyssey
Developer's Manual

Developer's Manual

Table of Contents

I) 6AD Team:

Name	**Title**
Connor Williams	IT Manager
Sheldon Lockie	Software Architect
Lucas Jackson	Quality Assurance Manager
Brandon Foss	Project Manager
Jubal Mitchell	Software Architect (Coding Standards)
Joshua Dempsey	Documentation Specialist

II) Environment Requirements

Obstacle Odyssey is currently being developed and designed to run on Unity version: 2018.3.11f1. While it may be compatible with other versions, stability and performance may be uncontrollably suboptimal. If you do not have Unity 2018.3.11 follow these instructions:

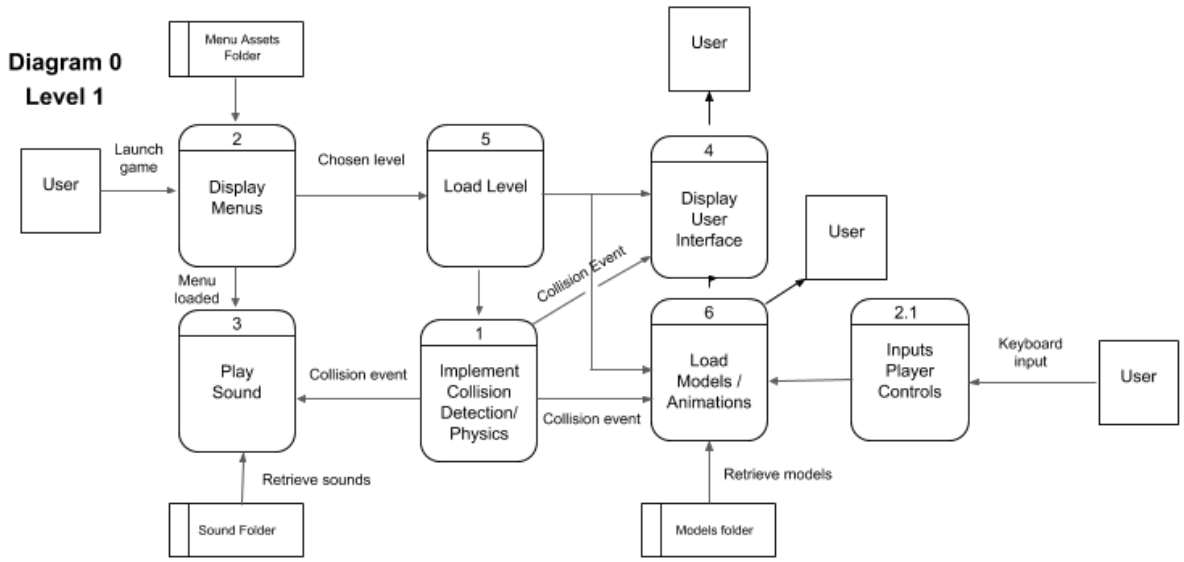
II a) Environment Setup Instructions

1. Navigate to https://unity3d.com/get-unity/download/archive?
    a. Make sure you are observing the Unity 2018.x patches tab
    b. Select the 'Unity Installer' download for the 2018.3.11 release from the 'Downloads' dropdown menu.
2. Open and run the installer
    a. Accept the terms and conditions
    b. Choose the Unity 2018.3.11f1 component

c. Select the desired directory for installation
d. Install Unity

III) Context Level Diagram:

The figure below is a Level 1 context Diagram 0 of the overall components and subsystems Obstacle Odyssey uses.

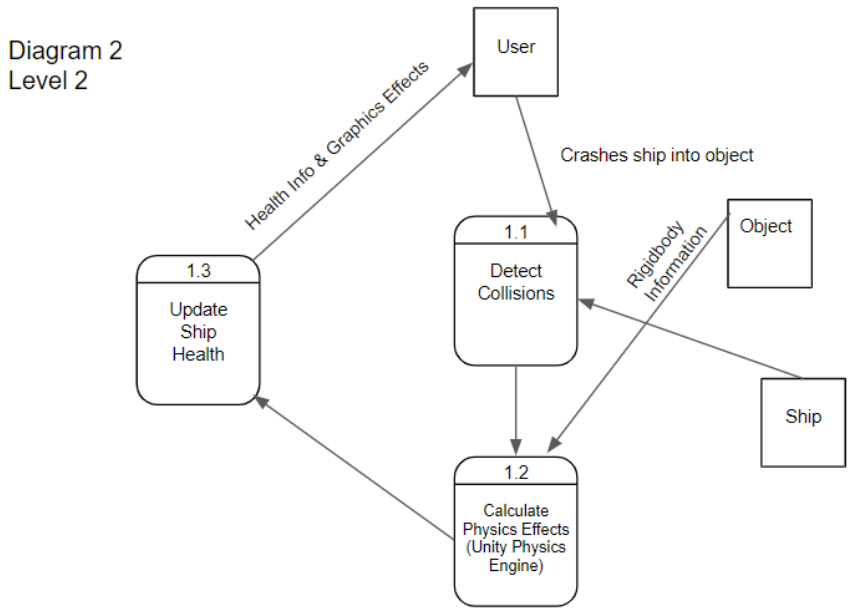


Below are several figures containing the context diagrams for the sub Systems and their respective 6AD developers.

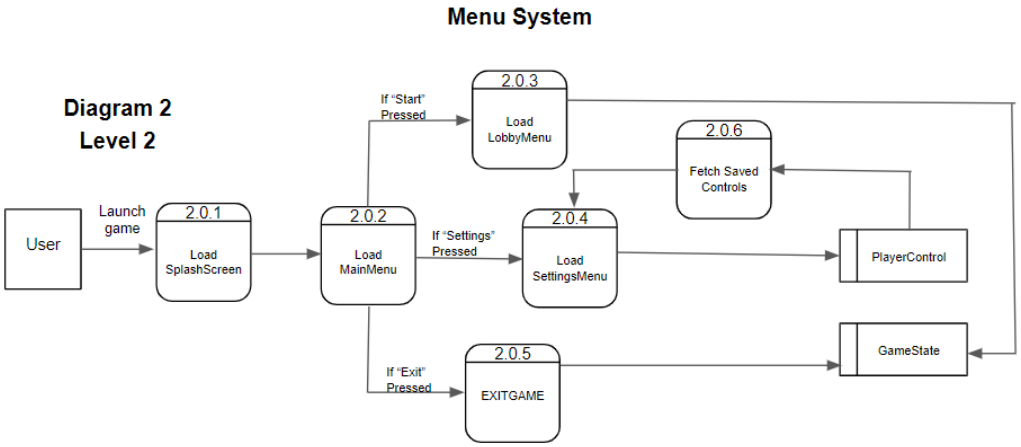
The owners and developers of each Component subsystem are as follows:

Name	**Developer**
1) Implement Collision Detection/Physics	Connor Williams
2) Display Menus 2.1) Input Player Controls	Sheldon Lockie
3) Play Sound	Lucas Jackson
4) Display User Interface	Brandon Foss
5) Load Level	Jubal Mitchell
6) Load Models/Animations	Joshua Dempsey

III.1) Collision Detection/Physics sub Diagram

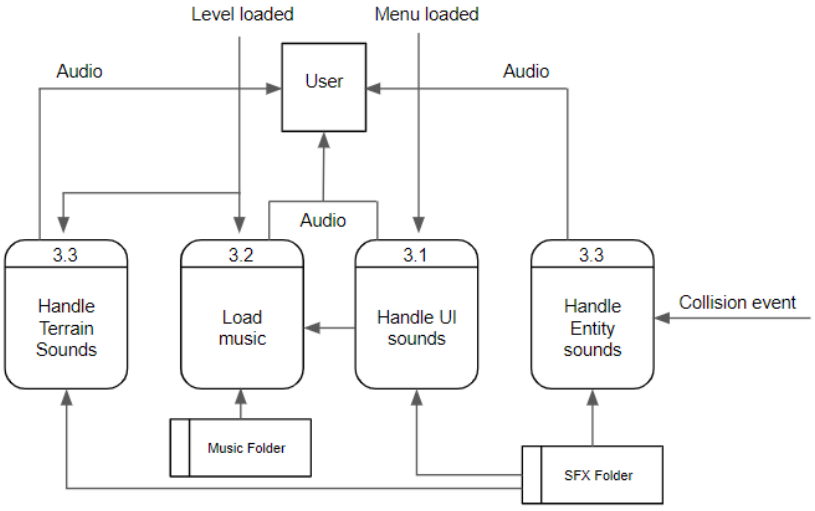


III.2) Menu System sub Diagram

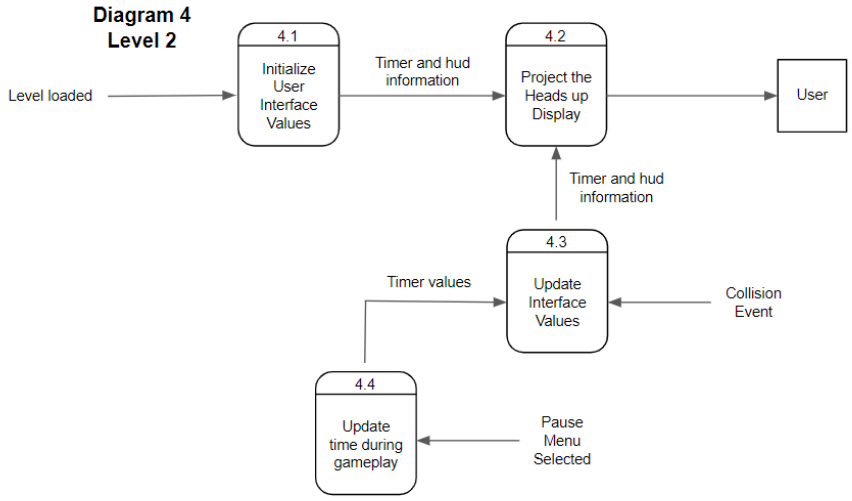


III.3) Sound System sub Diagram

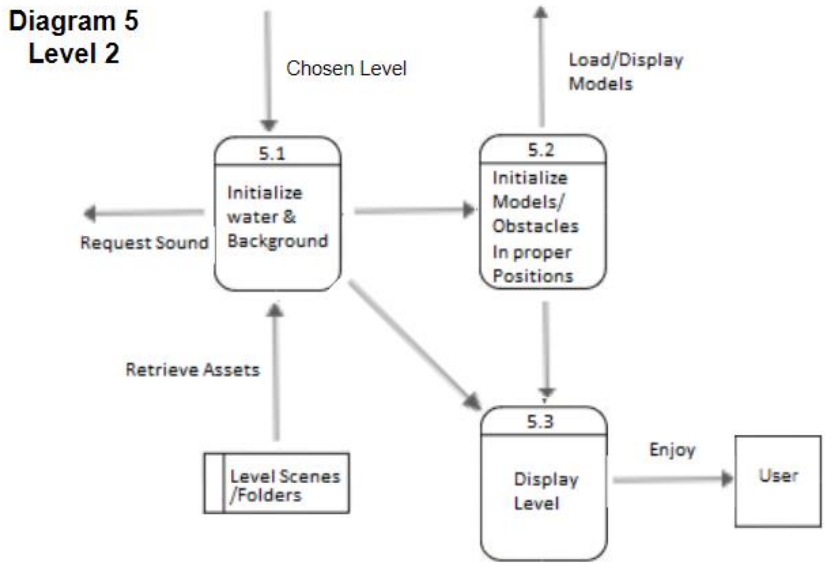
Diagram 3
Level 2



III.4) UI System sub Diagram

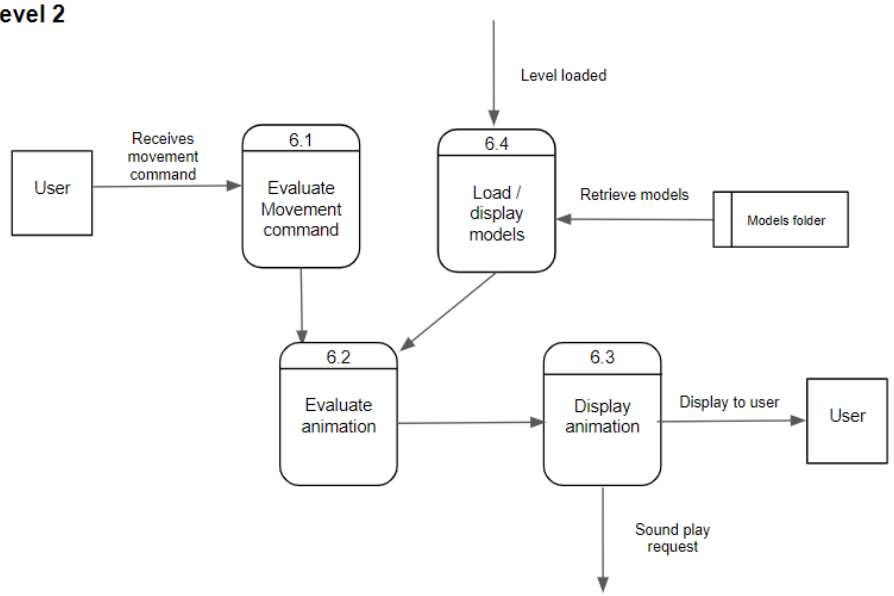


III.5) Level Loader sub Diagram



III.6) Model/Animation System sub Diagram

Diagram 6
Level 2

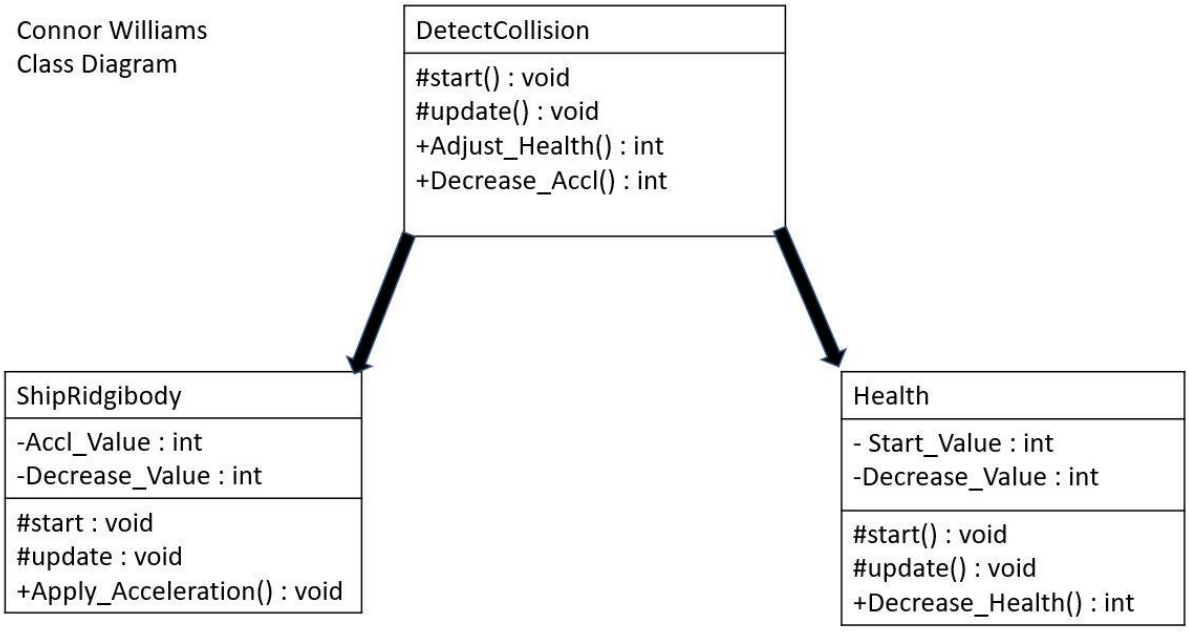


IV) Class Diagrams

This section contains figures and descriptions of class diagrams representing source code designed for Obstacle Odyssey.

IV.1) Class Diagrams: Connor Williams (IT Manager)

Diagram:

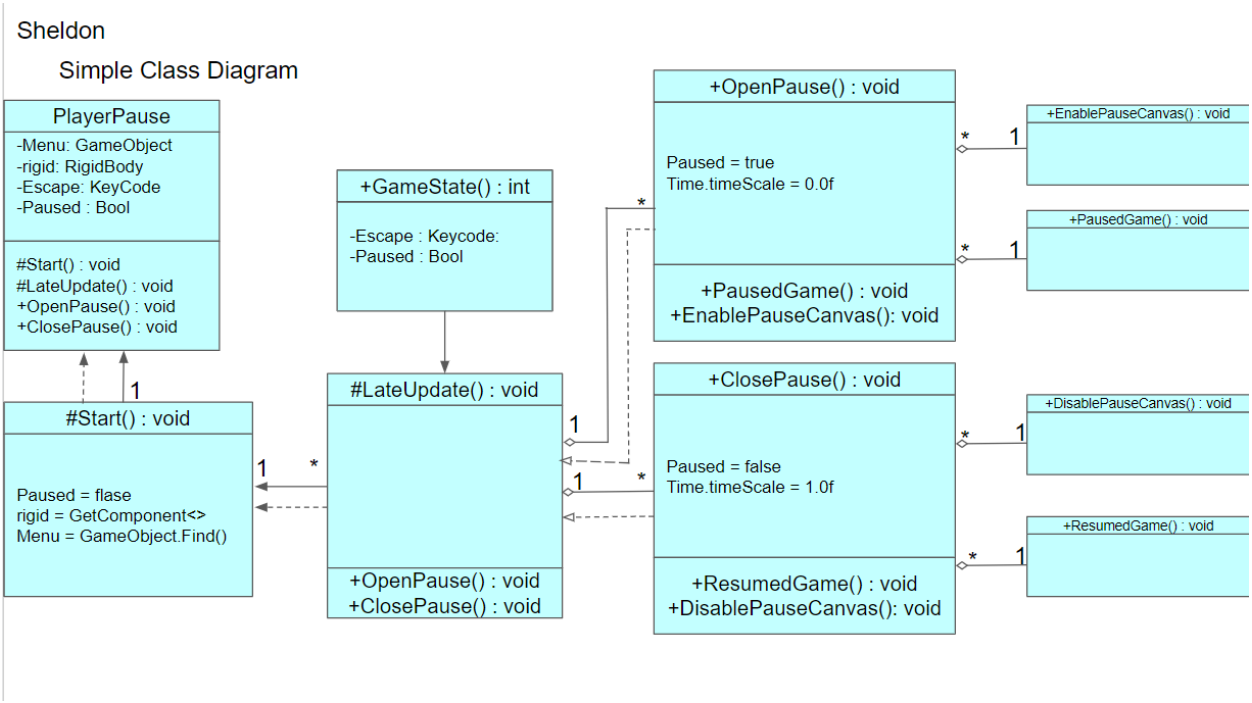


Description:

The main class, DetectCollision, utilizes Unity's Physics Engine to detect when the ship collides with GameObjects. Upon collision, it inherits from the Health class the instructions to perform on the ship's health. The RidgidBody of the ship is also updated in accordance with the collision.

IV.2) Class Diagrams: Sheldon Lockie (Software Architect)

Diagram:

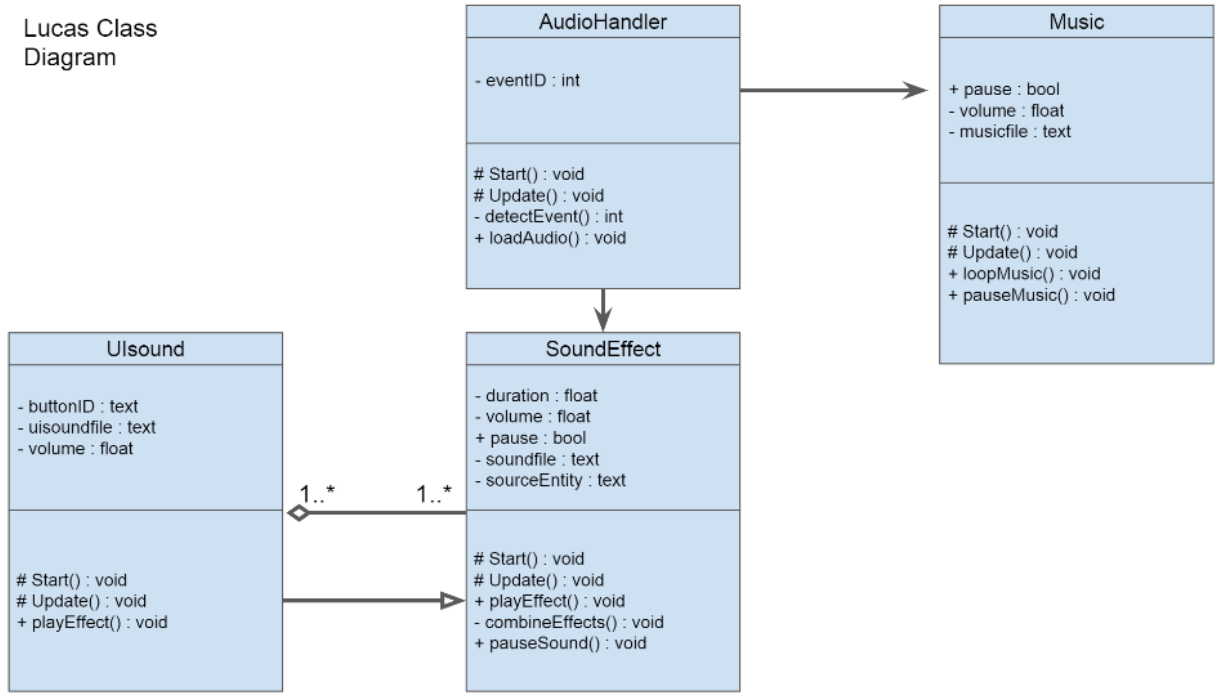


Description:

The above class diagram represents the relationship between classes in the pause menu, The General process is that Start declares all the public variables. On LateUpdate the process will look for an “esc” keycode, along with checking the current GameState, Depending on GameState, it will call OpenPause, stopping physics, and changing the GameState, or ClosePause, resuming physics and Changing the GameState. After OpenPause is ran it will invoke the EnablePauseCanvas class that will instantiate the menu, and PausedGame which will stop timers. After ClosePause is ran it will invoke DisableCanvas which will deinstantiate the menu, and ResumedGame, which will restart all the timers.

IV.3) Class Diagrams: Lucas Jackson (Quality Assurance Manager)

Diagram:

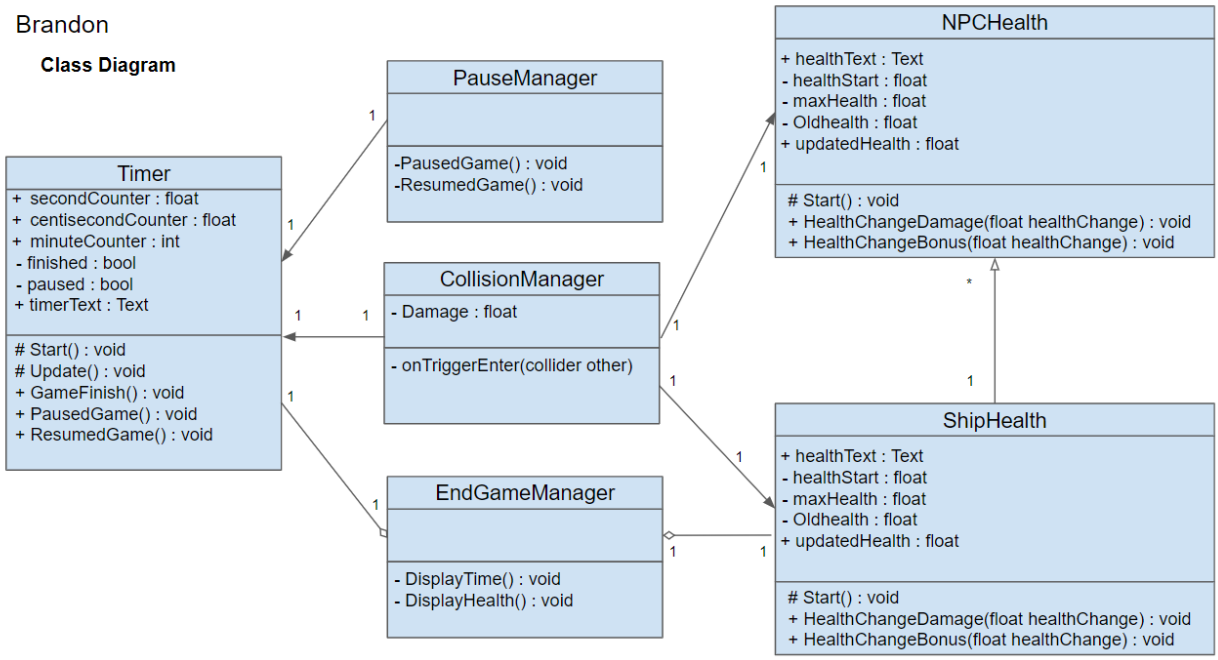


Description:

The audiohandler class is a singleton that can bind audio to any object in the game. It is able to call music, sound effects, and user interface audio, each with their own properties, depending on the audio file requested. Any audio file can be set to loop upon playing, as is the case for the background music and ambiance sounds in the game, and the volumes of each clip can be set individually.

IV.4) Class Diagrams: Brandon Foss (Project Manager)

Diagram:

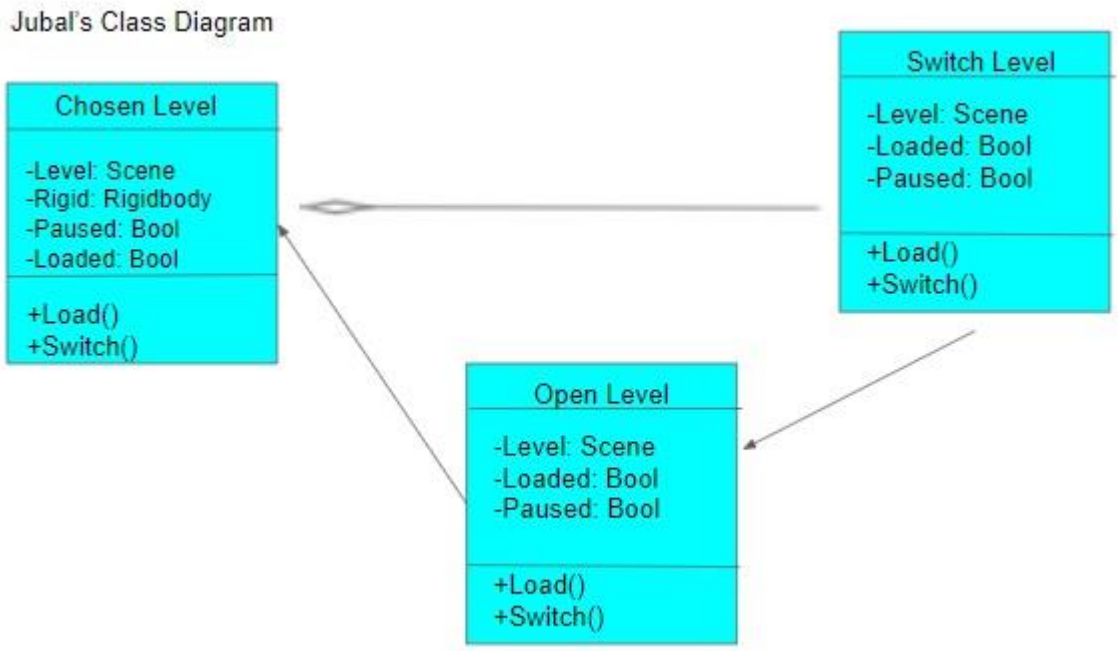


Description:

The diagram above represents the HUD that is displayed throughout gameplay. For the health field, a health class is created to track the max health and any changes that occur through interactions with the collision manager class. This can be inherited from to implement an enemy health class that takes the core features of the original class and modifies to the specific enemy. The timer class tracks time when the level loads and counts from 0. It interacts with the pause manager, collision manager, and end game manager to correctly alter the time values.

IV.5) Class Diagrams: Jubal Mitchell (Software Architect (Coding Standards))

Diagram:

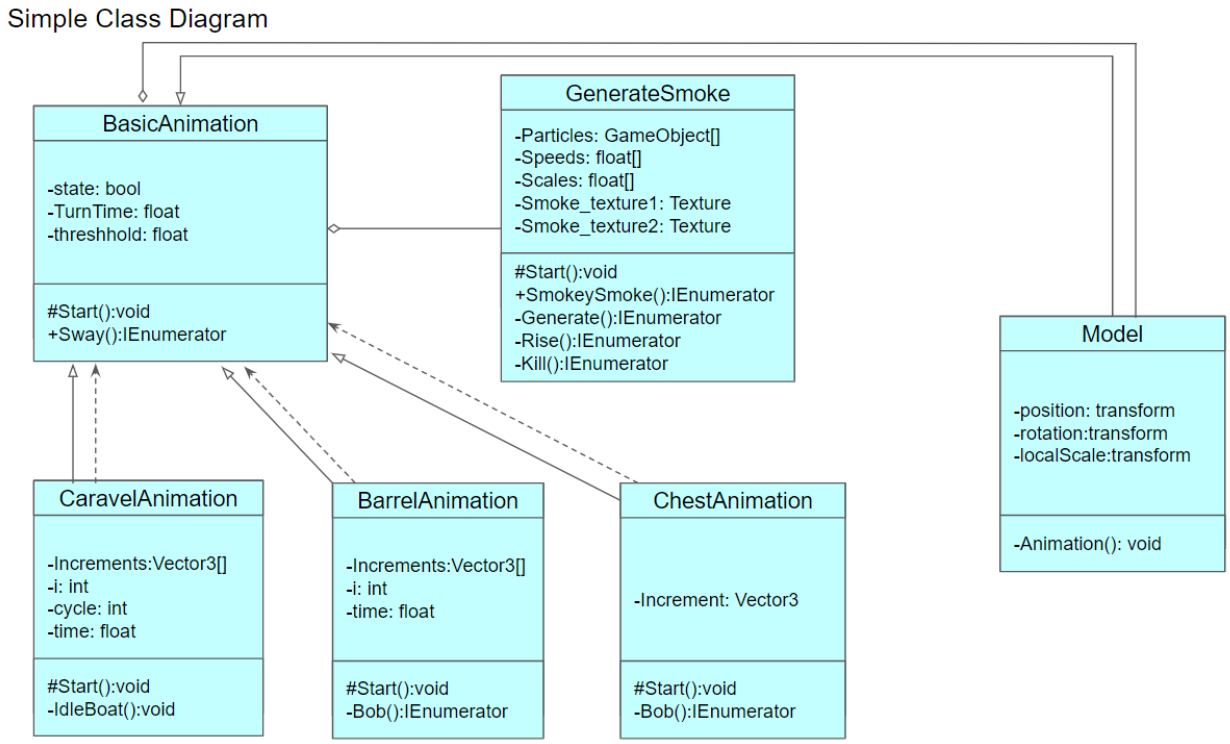


Description:

The class diagram for the level design/loader feature, originally started with different cases to load, open, switch between scenes, and instantiate models as needed. The class diagram represents the classes associated with these different options, however as my understanding of Unity increased and I became more familiar with how it works. Using the Unity Scene Management library, it was very simple to switch, load, and open new scenes, and therefore still followed the concept of the class diagram and was implemented slightly differently. Furthermore, since it was simpler than anticipated, the person in charge of this feature has worked on other things, such as whale movement, and will still be designing a level.

IV.6) Class Diagrams: Joshua Dempsey (Documentation Specialist)

Diagram:



Description:

The Figure above describes the relationship between basic animations and specific object animation sets. An object inherits simple animations from the superclass BasicAnimation and contains more object specific methods are contained in their ObjectAnimations' respective scripts. A Unity Model/GameObject has its corresponding animation subclass attached.