Bubby’s Deep-Sea Odyssey:

Development Documentation

Introduction:

Bubby’s Deep-Sea Odyssey is an educational exploration/puzzle game where the player must use 3 types of sensors to help the anglerfish Bubby navigate the dark depths of the ocean while trying to find his way back to his home in the deep trenches. The first level covers the use of the LIDAR sensor, what it can detect and where it falls short. The second level introduces the SONAR sensor and helps the player learn about what SONAR can do and how it differs from LIDAR in range, what it can sense, and where it falls short. The final level is the Camera level, which introduces the ability for the player to increase the light around Bubby so they can see their surroundings, mimicking the function of a camera in AI systems. This final level puts players to the test, using all three sensors to find the exit and finally reach Bubby’s home and the end of the game.

General Notes:

* Code:
  + All scripts associated with the project are enclosed in the Scripts folder in the project. The PlayerMovement and Sensor scripts are the most likely to need edits/changes, as they control functions like the sensors, stamina/energy levels, movement and more.
  + Each sensor has its own script that controls its use; they can be found in the SensorScripts folder within the larger Scripts folder.
    - Note that the LIDAR sensor is controlled by the script RadarControl; this naming is an artifact of when the LIDAR sensor was originally created to be a RADAR sensor. We switched when we discovered that RADAR doesn’t work well underwater and replaced it with LIDAR and SONAR.
    - The ObjectPooler script handles creating & maintaining the correct number of particle systems for the LIDAR sensor but is not included in the sensor folder; unless you’re doing a complete rehaul of how that sensor works, you shouldn’t need to touch this script.
    - The SONAR sensor relies on two scripts; the SonarPulse script handles most of how the SONAR sensor works, but the SonarPing script handles the actual markers that show up where the SonarPulse detects a collision.
  + All dialogue in the game uses the Yarn Spinner package; there are extensive online tutorials about how to use this dialogue system. All the scripts associated with Yarn are in a specific folder in the Scripts folder. The obstacles in the scenes are all set up with the Yarn Interactable script to make them trigger dialogue when Bubby runs into them – look at those obstacles to see how they’re set up if you want to add more obstacles/dialogue options. To write more dialogue, edit the Intro\_Narration yarn script. To ensure that the dialogue plays correctly, there must be a Dialogue System and a DialogueAdvanceInput in the scene. The Dialogue System also defines how the text shows up on the screen- if you want to edit the appearance of Bubby’s dialogue, you’ll need to make edits to that game object.
    - Note that when testing the Sonar or Camera scenes in Unity, you’ll get an error on line 44 of the Yarn Interactable script if you run into an obstacle or object that triggers dialogue – this is because it needs the DialogueTracker script somewhere in the scene, and it lives in the Radar scene and carries over from scene to scene recording what dialogue nodes the player has already seen. So if you run into this error don’t panic it’s normal, just start playing and progress normally through scenes starting with the radar scene or main menu and it’ll work fine.
  + All scenes/levels can be found in the Scenes folder. The player progresses from one scene to another based on the order set out in the Build Settings (currently Main Menu, RadarLevel, SonarLevel, CameraLevel, Homecoming, then loop back to the main menu)
    - Note that the RadarLevel is the first level where the player uses the LIDAR sensor; the name is an artifact of when that level was originally going to be for the RADAR sensor, before we discovered that RADAR doesn’t work well underwater.
    - There is a folder for test scenes in here too, but they don’t contain anything important, just some spaces to play around with new sensors/functions.
  + Note that all scenes require the LevelLoader to function properly, and all scenes but the beginning and end scene require the Dialogue Tracker, which is a singleton object that tracks which dialogue options the player has already seen so they don’t see repeats.
  + All sensor levels have a Background, Midground and Foreground that create the environment/maze, but remember that due to lighting the player probably won’t see much of the environment at a time. Décor is important, but don’t spend too much time on it! Additionally, the player can only run into/interact with objects placed on the Foreground level.
  + The Player is split into multiple objects housed under a general PlayerMovement object for animation purposes, but also to handle making the sprite flip, appear on the minimap, making the light grow and shrink, etc. Do not change the order/layering of any object in the Player group without making sure it won’t mess up the code in PlayerMovement or any sensor script (and possibly others).
* UI:
  + The main UI is contained in a canvas within the main camera – this includes the minimap UI (though it has its own camera, you shouldn’t mess with the minimap camera unless you need to change something about the minimap – just change anything that needs to show up on the minimap to the Minimap layer) and the stamina/energy bar.
  + The LevelLoader has a canvas element that contains the black foreground image that makes for a smooth transition between levels. This is also where the Main Menu’s UI is located.
* Animations:
  + All animations are held within the Animations folder. All timeline and playable direction assets are in the Timelines folder in My Assets, and all the signals for the timelines are in the Signals folder in My Assets.
  + All levels include a animation controlled by the Timeline. The Main Menu has a credits animation (this is the only one that does not play on start) that is triggered by hitting the Credits button. All other animations begin as soon as the scene loads, and for the three sensor levels each animation functions as a tutorial introducing the new sensor added in that level. The Homecoming scene’s animation shows Bubby’s triumphant return at the end of the game and will automatically return the player to the Main Menu once it is complete.
  + Note that if you want any timeline other than the Main Menu to not play on start, you’ll need to disable the checkboxes for play on start/awake in the timeline object and add a blank state to the animator component of every animated object in the scene! It’s annoying but if you don’t add a blank state to those animators and make them the default state, they’ll just start playing the animations when the scene loads, even if the timeline object itself is disabled.
* Audio:
  + All background music and sound effects are controlled by the AudioManager object in each scene, which has an AudioManager script attached. An example of how to access/use it can be seen in the PlayerMovement script or the YarnInteractable script.
  + The sound files themselves are taken from Freesound.org (see the actual sources in the credits) and can be found in the Audio folder in the My\_Assets folder.
  + All levels have a specific background music (except for the main menu, which shares its background music with the sonar level), some underwater ambiance that never turns off, and an SFX audio source that only plays a sound when prompted (see the AudioManager script)
* Art:
  + The project includes a 2D Hand-painted underwater assets pack and a rounded UI pack, which was used for most of the art/assets.
  + Any assets not part of this pack can be found in the My\_Assets folder. You can also find prefabs in here, like in the Obstacles folder. Most notably, the Signpost and Bubby assets can be found in this folder.
  + Most if not all text is in the Riffic Free Bold font – note that there are two versions of that font in the project and only one of them works (I’m not sure why, and I can’t figure out which one is the problem one to delete/remove it).

Future Work:

* Potential Changes to make in the future:
  + Make the Bubby sprite flash red when it hits something
  + Add a reset button to reload a level if you get stuck (just reset the player position to the beginning of the level, don’t reload the scene or it will trigger the intro animation again)
  + Implement Challenge Mode:
    - Reload/restart scene after you hit no stamina in the energy bar 3 times (possibly indicate lives with battery icons next to the stamina/energy bar)
    - Could make it so you have more “lives” in earlier levels and less in later levels
    - Maybe find batteries around the level to refuel energy/get more lives?
    - Addition of a timer to help increase urgency (the original concept had a shark chasing Bubby in later stages – this could be another option)
  + Add treasure to points count or # treasure found count
  + Change the dialogue box so that there’s an actual dialogue bubble around it – make it look like a bubble too! Some of the playtesting feedback I got a while back said it would be nice if the dialogue appears to be spoken by Bubby.
  + Check on the learning goals - make sure the game includes all of them and make sure the learning is baked into how the game works. Basically, improvements to the learning content.
  + Add a little bloom/glow effect around Bubby’s dangling light

Link to all playtesting feedback: <https://docs.google.com/document/d/1ZFUn4dSAyzbl2uhyzUmQCCUGj8h99tNBHhJwwn16KOI/edit?usp=sharing>

Link to original slideshow: <https://docs.google.com/presentation/d/1fIhwj_FQd84AJTDjXdGo4fBJUVwjD0XE7oFMIJVg4w4/edit?slide=id.p#slide=id.p>