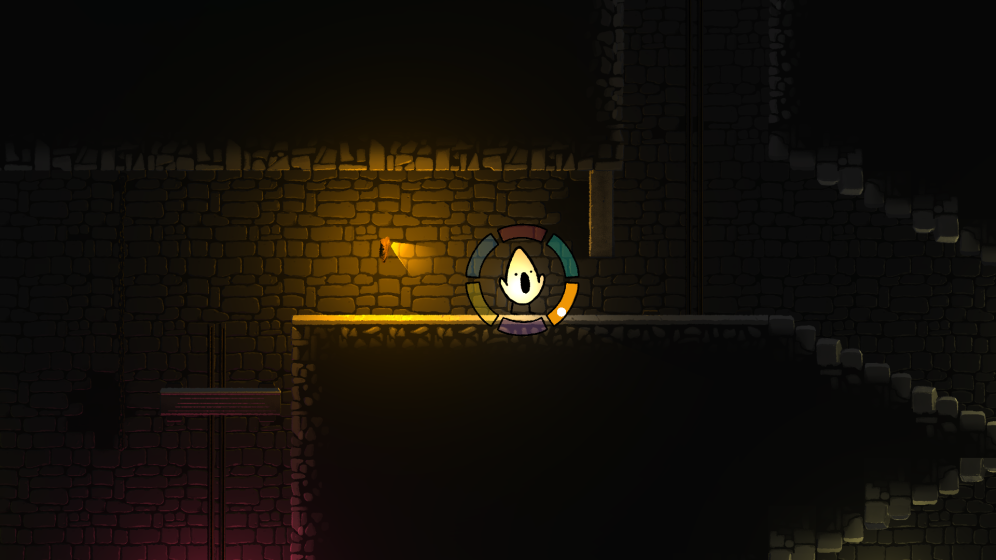
Lullaby

Factsheet



Lullaby is an atmospheric 2D puzzle platformer where you use song to overcome the challenges of a mysterious dungeon.

**Team**

|  |  |  |
| --- | --- | --- |
| **Team member:** | **Primary role:** | **Secondary roles:** |
| Martin Emil Hansen | Design Lead | Game Design Lead, Art Lead, Level, and Sound Design, Lighting |
| Mark Staun Poulsen | Producer | Game Design, Level Design, Sound Design |
| Jonathan Emil Mogensen | Tech Lead | Gameplay Programming, Level Design, Lighting |
| Magnus Nakskov Laursen | QA Lead | Game Design, Level Design |
| Marcus Grosen Poulsen | Programmer | Gameplay Programmer, Level Design, Audio Integration |
| Fernando Rodríguez Álvarez | Programmer | Input Designer |

**Summary**

Lullaby is a 2D puzzle platformer played using a modern Xbox or Dualshock controller. In the game, you play as a triangle who can manipulate the world around it through song. To do this, the player interacts with an in-game UI-wheel using the controller to select a note from the wheel and play it by pressing a button. This selection of notes is then used in various combinations and sequences to solve puzzles throughout the three levels. The target audience is people with no prior musical training and the game is designed with colourblind people in mind as well.

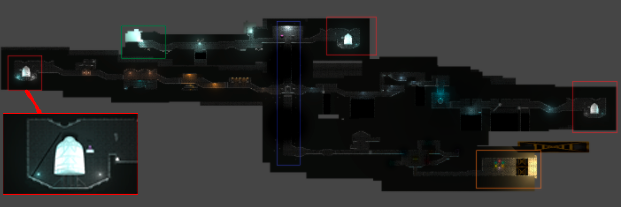
**Gameplay Setup**

The game is played on a PC running Windows. It is a single-player game and requires a controller in order to be played.

* **Controls:**
* Left stick: Move character
* Right stick: bring up the singing wheel, select note
* Left/Right bumper: lock note
* Left/Right trigger: sing note (chord sensitive)
* X/A: Jump

**Levels:**

The level progression happens in a semi-linear fashion, where the player starts (green square), solves the puzzles to ring a bell (red square), and progresses to the next area (blue square). The exception to this linearity is the middle level, where you can complete the left or right wing in the order you like. The game is finished at the orange box, and thus has a finite length.



**Core mechanics**

The core mechanics of the game are:

* **Singing**: the core mechanic of the game. The player can sing one or multiple notes at the same time in order to interact with the world. Different elements of the world are interacted with differently, but all of them require singing. Most of the interactable objects have a visual cue signifying what should be sung. The volume of the notes can be modulated by the pressure on the controller trigger.
  + **Note locking**: Different notes can be locked at the same time, so when the singing starts, all of the notes selected will be sung at the same time.
* **Movement**: the player can control the character in 2D-space, using walking and jumping in order to navigate through the dungeon.

**Main systems/elements**

Interaction Architecture: The main code architecture for interacting with objects in the game. It utilizes a strategy pattern to separate how objects receive input and how they behave depending on the input.

NoteWheel: The main UI-component that shows the player what they can sing, and thus supports interacting with the world.



**Game Loop(s)**

Enter room -> use song to solve puzzle -> keep exploring

**Look and Feel**

You control a cute and expressive character navigating through an increasingly dark and atmospheric dungeon. The dungeon has a hollow hum ringing through the various rooms, but your character is able to bring levity and life to the environment through their beaming song. The game features a hand-drawn 2D art style and uses wholly originally crafted visual assets. Sound effects, such as ringing bells and grinding doors are edited and used to emphasize key game moments and build the soundscape.

**Technical information**

Engine used: Unity 2019.4.8f1

We made use of Unity’s Universal Render Pipeline, in order to apply 2D lighting to the game. We also made use of Unity’s new Input System v1.0 to add better controller support.

We have also made use of the FMOD Studio audio engine for the implementation of all the sounds in the game. FMOD allows for more precise sound tweaking and experience. We have made use of FMOD’s plugin that allows linking FMOD projects with Unity projects.

**External Assets:**

**Unity Assets:**

- **DoTween**: <http://dotween.demigiant.com/index.php>

- <http://dotween.demigiant.com/license.php>

- **FMOD**: <https://www.fmod.com/>

- <https://fmod.com/licensing>

**Sounds:**

Several sound files used in this project come from Soundly’s library of sounds

- <https://getsoundly.com/assets/Soundly-EULA.pdf>

- **Rumble**: “rumble.wav” by tim.kahn <https://freesound.org/people/tim.kahn/sounds/94114/>

- <https://creativecommons.org/licenses/by/3.0/legalcode>

- **Door**: “<a href="http://OGsoundFX.com" rel="nofollow">OGsoundFX.com</a> Free Pack” by OGsoundFX <https://freesound.org/people/OGsoundFX/sounds/423146/>

- <https://creativecommons.org/licenses/by/3.0/legalcode>

- **Wind Chime**: “Wind Chime, Gamelan Gong, A.wav” by InspectorJ <https://freesound.org/people/InspectorJ/sounds/411090/>

- <https://creativecommons.org/licenses/by/3.0/legalcode>

- **Chain loading bay door**: “ChainLodingBayDoor01.WAV” by mmaruska <https://freesound.org/people/mmaruska/sounds/241108/>

- <https://creativecommons.org/licenses/by/3.0/legalcode>

**Fonts:**

Indian Type Foundry - Kalam <https://fonts.google.com/specimen/Kalam?query=kalam#standard-styles>

* <https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL>

Vernon Adams - Amatic SC <https://fonts.google.com/specimen/Amatic+SC?query=amatic#glyphs>

* <https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL>