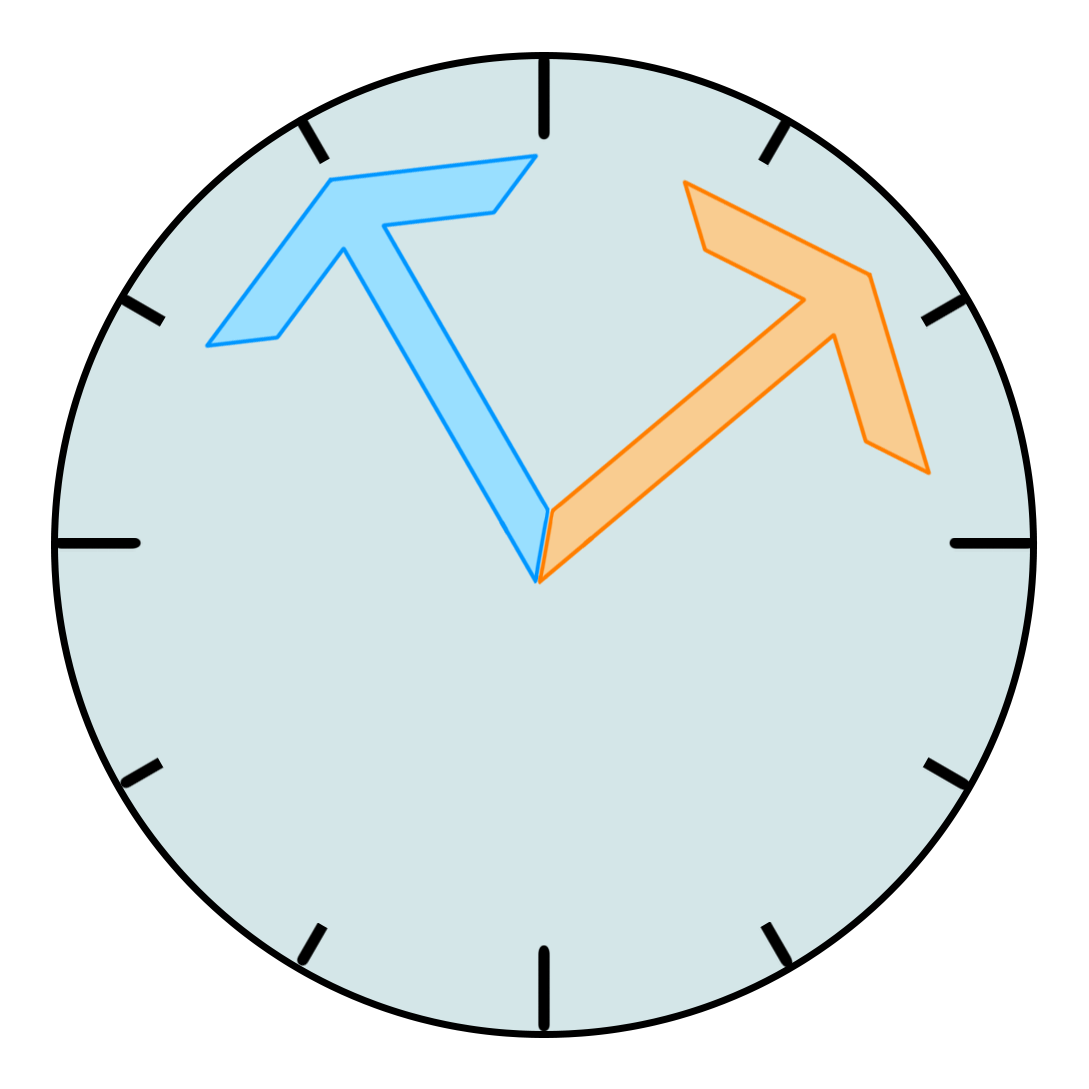
**Time Tested**

**Members:** Itay Bernstein & Avital Gordin

**Advisor:** Moshe Sulamy

**Workshop:** Game Development with Unity



**Summary**

The players are volunteers for a lab experiment run by 2 scientists studying the possible uses for time manipulation. The experiment puts the players' problem-solving abilities to the test and allows them to use gadgets developed by the scientists to progress in the game. This game is all about solving puzzles, defeating enemies, and clearing different stages in a multiplayer setting. The multiplayer aspect of the game will run on a cloud service. Many of the game's mechanics can be controlled by voice commands.

**What Makes the Game Unique**

The unique aspect of the game is the ability to manipulate time in a multiplayer setting and use voice commands to activate said time-related powerups. These powerups are used to manipulate and complete the game's levels. The game will also include narration dialogue that will help immerse the players into the plot of the game and make it a fun experience. Thanks to the multiplayer features, players will be able to invite their friends to join their game and cooperate through different puzzles together.

**Similarities and Differences with Other Existing Games**

A few games use similar ideas and features that allow the player to manipulate time. One of them is a game called Braid, which was released in 2008. However, the game has a few differences compared to our idea; Firstly, the game is in single-player mode. In addition, when rewinding time in Braid, all objects on screen, including the player, are affected. In our game, certain objects will be interactable using time-manipulation and other power-ups. Lastly, the plot of Braid is entirely different from ours, while our game is about scientists using the players as lab rats for experimentation, Braid is an adventure game about saving a princess from a monster.

**Main Project Features:**

**“Time-Manipulable Objects”:** The game will feature objects in levels that are affected by time manipulation power-ups (will be elaborated on below). When such power-ups are used, all time-manipulable objects on the screen will be affected. Players are not time-manipulable.

**Enemies:** Some levels will have enemies in them that will be hostile towards the players and try to attack and kill them. All enemies are considered “time-manipulable”.

**Power-Ups:** The game will feature power-ups that will be used to complete the puzzles.

Here are the various power-ups that will be scattered throughout the game:

* **Time Rewind:** When used, will rewind time for all time-manipulable objects onscreen (including enemies, as was mentioned), bringing them back to a past state and position.
* **Time Stop:** When used, will stop time for all time-manipulable objects onscreen. These objects will be frozen for a certain amount of time. Time Rewind will not take into consideration the positions and states of objects/enemies during Time Stop and will skip over them.
* **Superweapon:** The game will feature a “superweapon” power-up (undecided what kind of weapon it will be, but it will be destructive). This superweapon will be very effective against enemies and will be able to destroy time-manipulable objects.
* **Clone:** When used, the player will create a clone of themselves in their exact position. Can be used to keep buttons pressed. In addition, enemies will be distracted by the clone, targeting it instead of the players. The clone will have its own health bar. When the health bar depletes entirely, the clone will disappear. The power-up can be used again to destroy the clone manually. The clone is time-manipulable.
* **Swap Position:** When used, the player will swap position with a targeted object/enemy. The player will have to target an object/enemy beforehand in order to use it.

**Voice Commands:** All power-ups in the game will be able to be activated using voice commands. The voice command for Time Rewind will include a parameter, for example, "Time Rewind **five** seconds."

There will be an option not to use voice commands.

**External Dependencies:**

**Unity:** The game will be developed under the Unity game development engine.

**Voice-to-Text:** This project will use Deepgram’s API to send voice recordings and receive their transcript to facilitate voice commands.

**Cloud service:** This game will have its multiplayer component run on a cloud service.

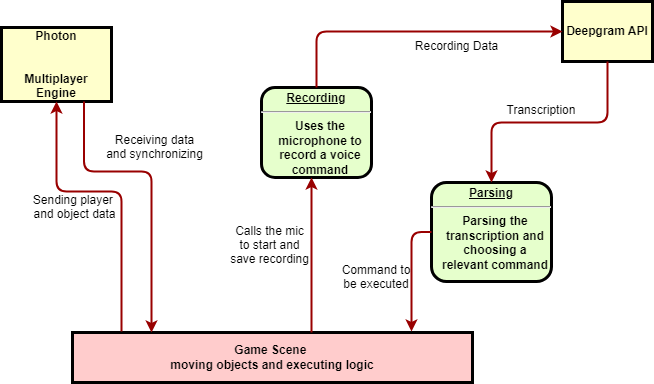
**Architecture:**

**Components:**

**Unity:** The game engine the game is developed on. Unity is a framework for video games, providing the ability to create, animate and perform logic on various objects in scenes. Unity is the core of our game and performs most of the game’s actions, overall brings the game “to life”.

**Photon Multiplayer Engine:** An engine for handling multiplayer. Unity will communicate with the Photon-PUN2 API to synchronize between players and objects in the game.

**Deepgram API:** The API used to analyze and transcribe the recorded audio in the game, turning it into text and sending it back.



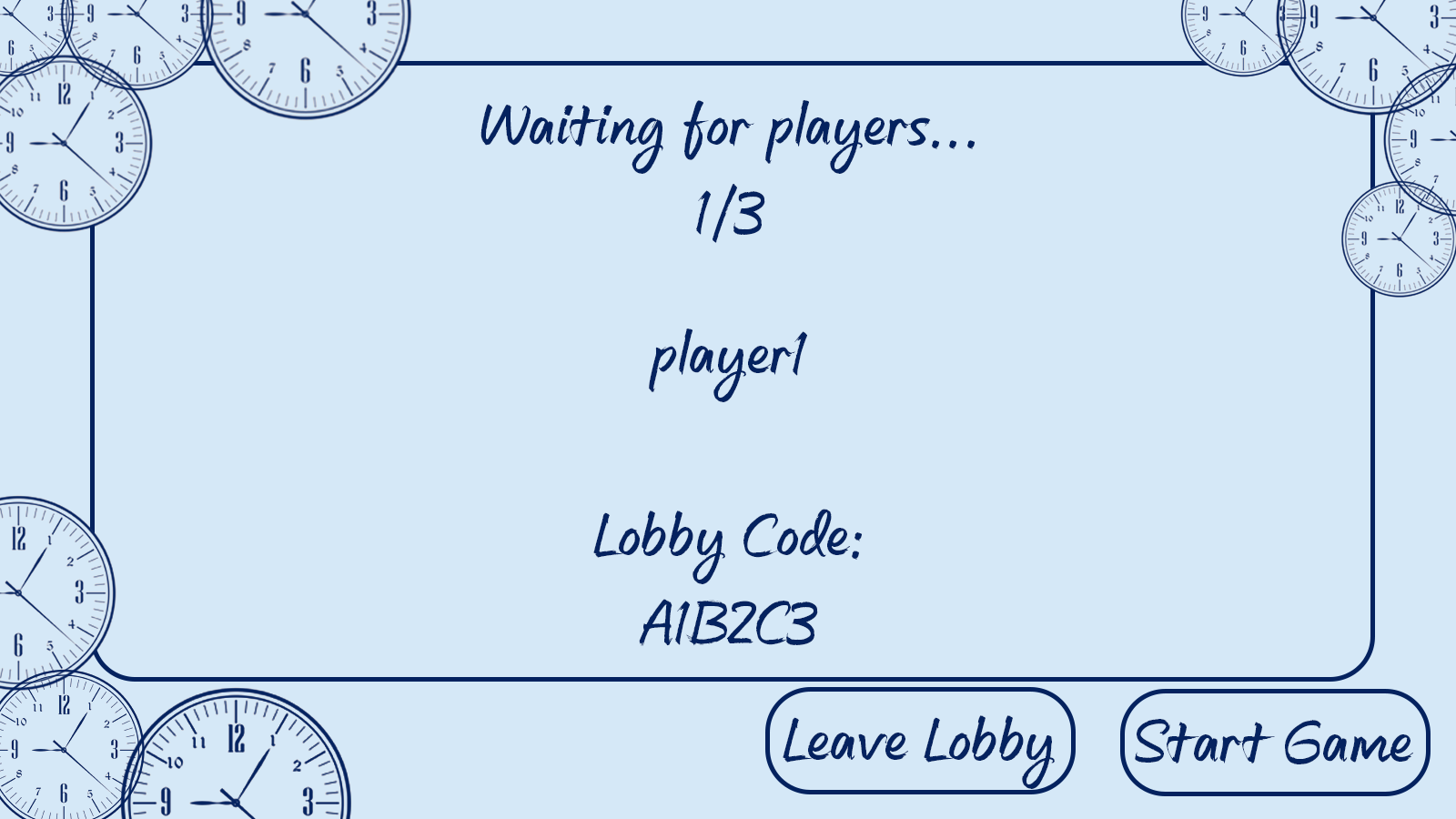
**Gameplay/Flow of Users in the Game**

The game is aimed at people who want to solve problems and puzzles, are interested in the concept of time manipulation, and want to cooperate and work together with other players.

Upon starting the game application, the player will be met with a main menu where they can join a game, change game settings, or quit.



There will be a lobby where players can wait for the game to start, accommodating up to 3 players.



After starting up the game, the players will be familiarized with the different mechanics, primarily showcasing the various power-ups available throughout the game’s levels. The 2 scientists will narrate them through the tutorial to explain each mechanic.



Afterward, the players will advance through different rooms/levels with puzzles and enemies in them and will have to use teamwork and the game’s mechanics effectively to clear them. The 2 scientists will occasionally interject with narration according to various game triggers.

To execute a voice command, the player will press the record button (“R” by default) and press the button again when they finish recording.

**Testing:**

**Voice Commands:** At first, tests were done to ensure compatibility with various voice-to-text APIs. Deepgram API was chosen, then further testing was done to make sure extracting the relevant text from the API response was working correctly.

Multiple tests have been done to measure the response time from Deepgram (around 3-4 seconds).

Then after parsing and connecting it to the relevant power-up (was tested on Time Rewind) further testing was done to make sure the power-up is executed correctly and that the parameter given inside the command was parsed and translated into the power-up effect successfully.

**Photon PUN2:** When work on the project has begun, two different Photon APIs were tested (Photon PUN2 and Photon Fusion). PUN2 was eventually chosen. The feature of connecting to the waiting room was tested, and after working, was implemented along with a connection to the main game scene.

**User Guide:**

**Using a power-up:** To use a power-up, the user needs to record a voice command. To record, the user needs to press the record button (“R” by default) once to start recording and then again to finish the recording.

The power-up relevant to the voice command will be executed.

In case of the Swap power-up, the user will need to target a time-manipulable object before using the power-up.

**List of voice commands:**

“Time rewind [parameter] seconds.”

“Stop time.”

“Use super weapon.”

“Clone.”

“Swap positions.”

**Maintenance:**

**Time-manipulable objects:** To add a new object that is time-manipulable, simply give the object the tag “Time Manipulable” in Unity. The object should then be affected by power-ups.

**Power-ups:** The power-ups are handled by a “Manipulator” object which executed power-up behavior and effects on all time-manipulable objects in the scene.

The Manipulator script has parameters to change the amount of positions saved each second per time-manipulable object and the max seconds in the past the Manipulator saves positions for.

**Voice commands:** Voice commands are handled in “VoiceCommands” script. This script handles using the microphone to record a command, sending the recording to Deepgram, and parsing the command. Adding new voice-commands and parsing for power-ups should be handled by that script.

Note: To test and play the game, a Deepgram API secret key is required. You can get one by signing up with a free account on the Deepgram website. Before updating the project on GitHub, you should remove any references to the key and replace them with a dummy version of a key.

This key is a parameter in the VoiceCommand script.

**Development Problems:**

**Lack of motivation:** Itay suffered from lack of motivation during the month of April. It characterized by barely going outside or doing project work while generally feeling depressed.

This feeling was overcame by forcing himself to start test various text-to-speech APIs and implement the voice commands. Once you start something, you’re more likely to finish it.

**Death of relatives:** The grandfather of Avital’s partner was hospitalized in the beginning of May and eventually passed away. This caused her to be occupied with hospital visits, the funeral, and comforting the grieving family throughout the month.   
**Surgery:** Avital had surgery at the end of May. This caused her a lot of pain and stress prior to the surgery, and she had frequent doctor visits, which further delayed her ability to work on the project.

Avital had to work extra hard to compensate for the two situations.

**Potential Future Work:**

**Adding more rooms and puzzles:** The project can be extended with more rooms with various puzzle layouts involving the different power-ups and objects, transitioning between 1 room and the next.

**Adding new power-ups:** The project can be extended by adding more power-ups to be used on time-manipulable objects, or perhaps power-ups that could affect the players.

**Adding new enemies and objects:** The project can be extended by adding different types of enemies with different behaviors. New objects (time-manipulable or not) can also be added to make new rooms with puzzles or spice up existing rooms.

**The definition of “Time-Manipulable”:** The definition of “time-manipulable” can be changed and expanded by making previously not time-manipulable objects into ones that are, adding new objects that are time-manipulable, or consider player characters as time-manipulable. Could also add different levels of definitions for time-manipulable objects that may be affected by only a subset of the power-ups, for example, an object that can be destroyed with a superweapon but cannot be manipulated by any other power-up.