**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 and other abilities. 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 and clearing different stages in a multiplayer setting. The multiplayer aspect of the game runs in the cloud. The power-ups are activated by using 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 also includes 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 navigate 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, only 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 features objects in levels that are affected by time manipulation power-ups (elaborated on below) such as boxes and spikes. When such power-ups are used, some or all time-manipulable objects on the screen will be affected. Players are not 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 are available in the game:

* **Time Rewind:** When used, will rewind time for all time-manipulable objects onscreen, bringing them back to a past state and position.
* **Superweapon:** The game will feature a “superweapon” power-up. This superweapon 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. The clone is time-manipulable.
* **Swap Position:** When used, the player will swap position with a targeted object.

**Voice Commands:** All power-ups in the game are activated using voice commands. The voice commands for Time Rewind and Swap Position will include a parameter, for example, "Time Rewind **five** seconds."

**Multiplayer Lobbies:** The game allows players to create and join rooms with a short code that can be shared with friends. The players will then play together in the same level.

**External Dependencies:**

**Unity:** The game was developed under the Unity game development engine.

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

**Photon Multiplayer Engine:** This game uses Photon services to run a multiplayer environment.

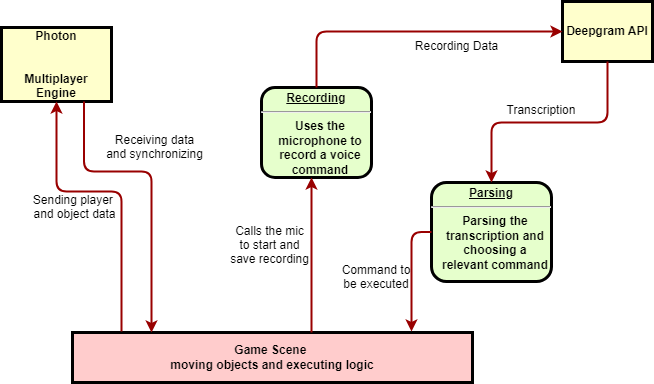
**Architecture:**

**Components:**

**Unity:** The game engine the game was 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 to the game client.



**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 and object manipulation, and want to cooperate and work together with other players.

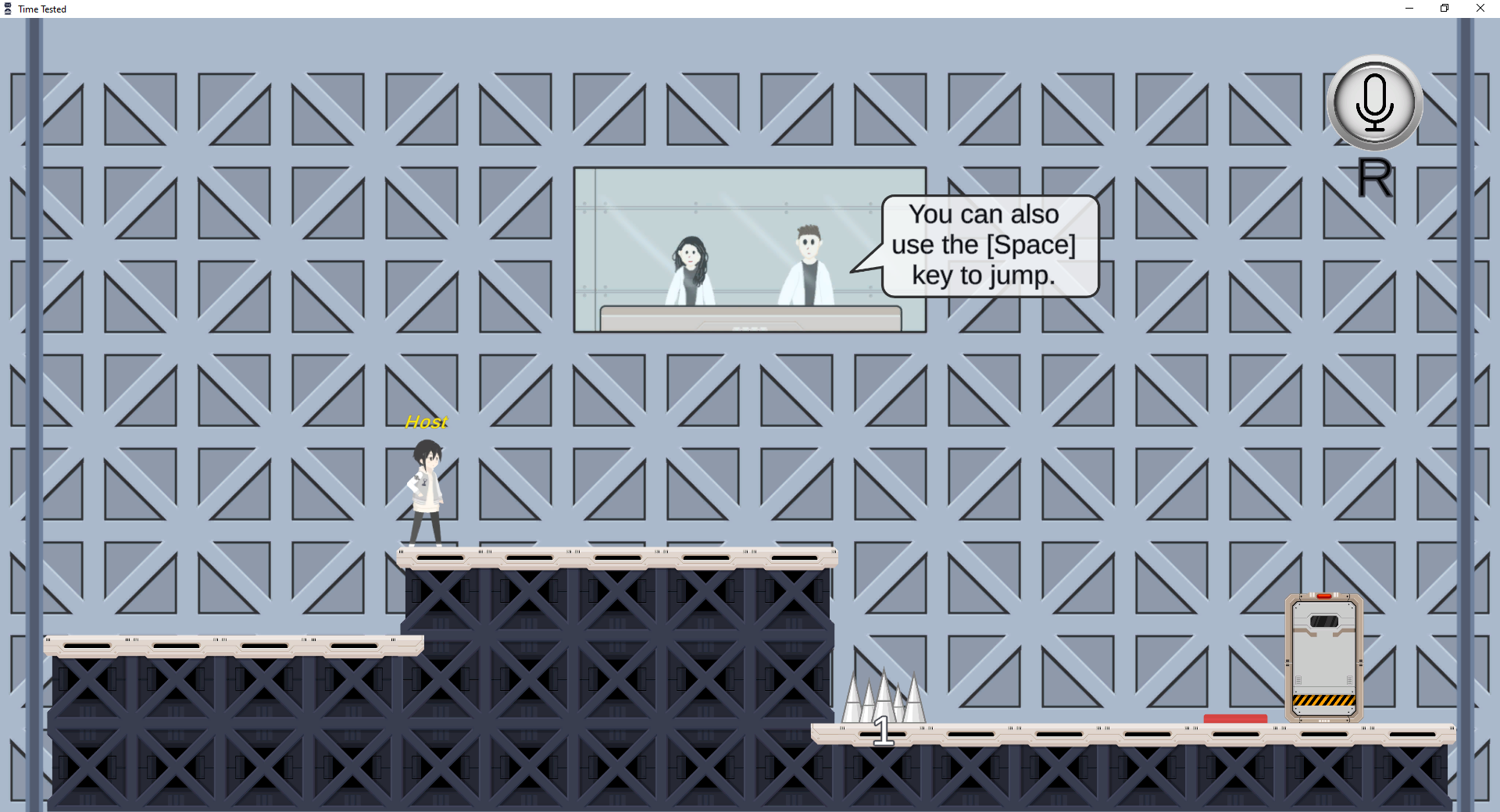
Upon starting the game application, the player will be met with a main menu where they can create/join a game, go to the tutorial, or quit.

****

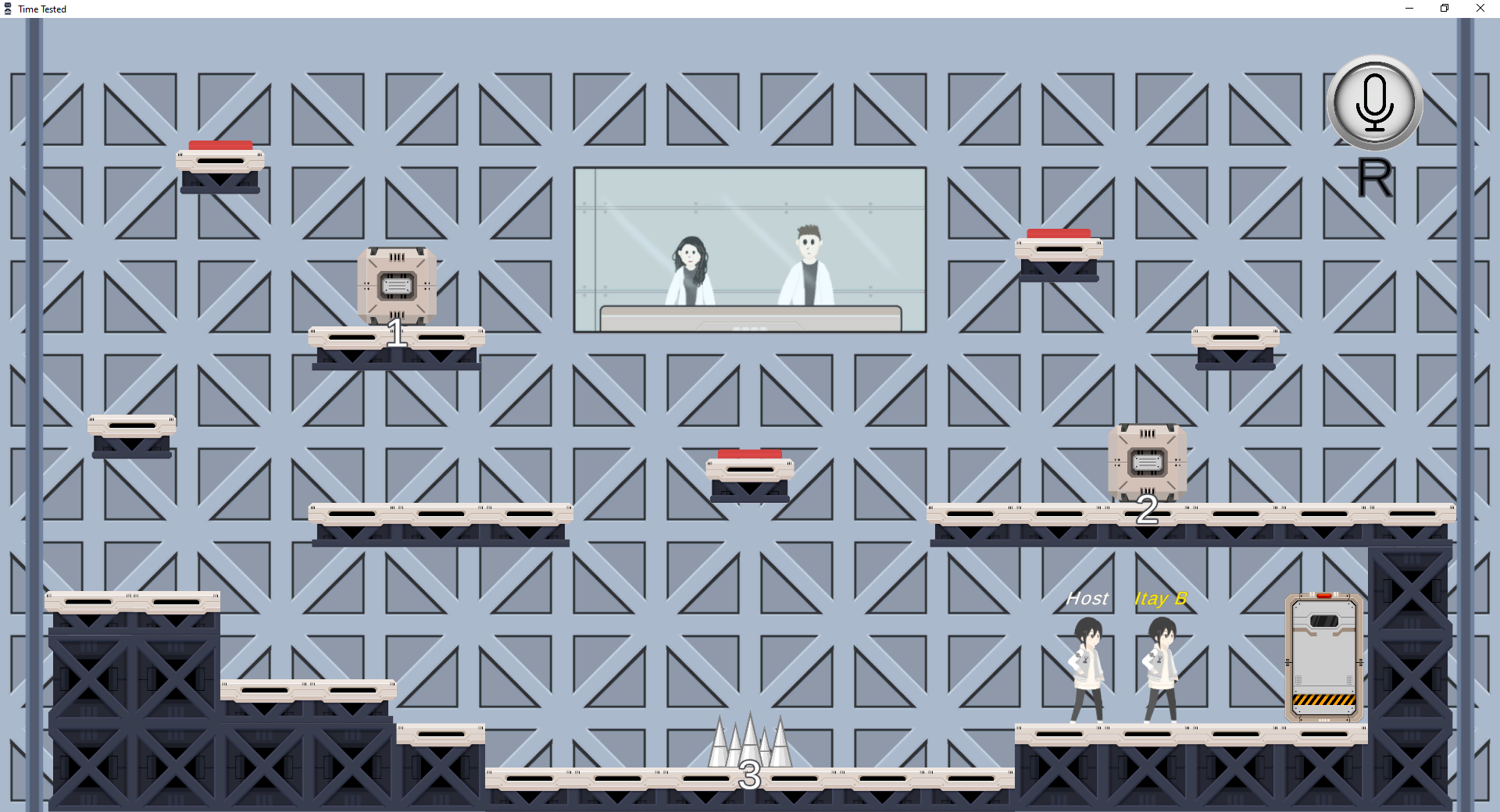
Upon creating a game, there will be a lobby where players can wait for the game to start, accommodating up to 3 players.



After starting up the game, the players will be familiarized with the different mechanics by going to the tutorial, 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 will have to use teamwork and the game’s mechanics effectively to clear them.



To execute a voice command, the player will press the record button (“R”) and press the button again when they finish recording, the game will then send the voice sample to Deepgram to analyze the voice command. After the analysis, the game will execute the power-up corresponding to the voice command.

**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 (up to 10 seconds in the worst case).

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.

**Full Integration Testing:** After successfully implementing rooms and multiplayer lobbies, many tests were done to integrate the game’s mechanics for use in a multiplayer setting and to ensure synchronization between clients as much as possible. This was done by playtesting and doing different actions in the game. Lots of bugs were discovered and knocked out by playtesting like this.

**Menus:** Tests were done in order to make sure the different menus in the game, such as the main menu and pause menu, were working as expected.

**User Guide:**

**Running the game:** Download the GitHub repository:

[**https://github.com/itaybrn/Time-Tested**](https://github.com/itaybrn/Time-Tested)

After downloading it, you can run the game by going to Project Files/Builds and running

**Time Tested.exe**.

**A complete tutorial on movement and using power-ups is available in the game. Alternatively, the instructions are detailed below:**

**Player Movement:** Use A to move left and D to move right. SPACE to jump.

**Game options:** Press ESC to bring up a game options menu, press ESC again to hide it.

**Using a power-up:** To use a power-up, the user needs to record a voice command. **A microphone must be configured to be the default microphone in order to use voice commands.** To record, the user needs to press the record button (“R”) once to start recording and then again to finish the recording.

The power-up relevant to the voice command will be executed after a short period of analyzing.

**List of voice commands:**

“Time rewind [No. of seconds] seconds.”

“Superweapon.”

“Clone.”

“Swap with [Object ID].”

**Maintenance:**

**Time-manipulable objects:** To add a new object that is time-manipulable, simply give the object the tag “Time Manipulable” in Unity, also make the object inherit from the class **HasID** and give it a textbox as a child in the hierarchy so that the “Manipulator” can ID it in the level. The object should then be affected by power-ups.

**Power-ups:** The power-ups are handled by the “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 number 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 (should be added to the **PowerUpCommand** class.

Note: To test and play the game, a Deepgram API secret key is required. We provided one included in this project, it is limited, thus we recommend replacing it occasionally. 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, under the object prefab of Player.

**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.

He overcame this feeling by forcing himself to start testing 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.

**Known Issues:**

**Syncing issues with bad internet:** The syncing between players can be delayed and choppy with bad internet.

**Non-host players can move boxes:** This is due to syncing issues in multiplayer.

**Voice analysis can take a bit long:** This heavily depends on Deepgram’s API and can’t do much about it aside from changing the API/library.

**Players can fall through the floor at very high speeds when falling.**

**Room code duplication:** There are 65356 possible combinations for room codes. When creating a new room, there’s a rare chance the generated room code already exists, and thus the create game button will not work and the user will need to click the button again.

**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.

**Adding sound effects and music:** The project could have various sounds effects for different power-ups as well as background music to the main menu and levels.

**Better multiplayer syncing:** The multiplayer aspects of the game can be rewritten to sync between players more smoothly.

**Fixing bugs:** Known and potentially newly discovered issues can be fixed to create a better and smoother overall experience.