

An Investigation of Correlations Between In-Game Location and Player Pitch

Vanessa Bolado, Nigel G. Ward
Computer Science, University of Texas at El Paso

Technical Report UTEP-CS-26-30

Abstract: For effective human-AI collaboration, systems need to be able to communicate effectively with their partner, including through use of appropriate prosody. While this will ultimately require detailed modeling of the user and their actions relative to shared goals, we here explored a possible shortcut: conditioning the prosody directly on an easily observable aspect of the current situation. Specifically, we investigated whether the pitch height of utterances could be predicted directly from the players' positions in a navigation-with-puzzles game. We developed a pipeline to extract, align, and visualize the relation between pitch and player position through heatmaps. There were some apparent spatial patterns. However, when we built a predictor and evaluated it in a leave-one-out experiment, spatial position alone was not a consistent predictor of pitch. This confirms that prosodic behavior is influenced by many factors beyond location.

Date: July 29, 2026

Correspondence: vboladogarcia@miners.utep.edu, nigelward@acm.org

1 Introduction

Human-AI collaboration is key to the wide deployment of future AI systems. If we wish to use AI systems to their maximum potential, they must be able to properly communicate with us. When we interact with other humans in dynamic environments, prosody plays a big role, and we wish to exploit this also in human-AI systems.

In general this is difficult. Many factors determine what prosody is appropriate in any specific situation, and various researchers are slowly elucidating what they are. At the same time, some aspects of prosody are sometimes strongly determined by simple factors, such as the social status of the addressee and the excitement level of the speaker. Here we set out to explore another simple factor, location, in the context of playing a cooperative online game.

Specifically, we examined whether the pitch of a player can be predicted from the current location in the game.

We chose to examine pitch, as it is relatively easy to calculate and relates to many emotional states and pragmatic intents. We chose to study this in a collaborative video game, *Fireboy and Watergirl*, where two players travel through a map filled with challenges they must work together to solve. We expected the pitch to be location-dependent because locations are associated with certain player states and activities. For example, we sometimes

observed the pitch of the guiding player increasing in regions of the map associated with obstacles, dangerous sections, or moments requiring close collaboration between players.

In general, the study of prosody in interaction involves two steps: first to explicitly model the relation between goals and situations and the mental states and communicative intents of the speakers, and second to model the relation between these and the prosody. In contrast, this work represents an investigation of a potential shortcut: whether an AI agent can produce more contextually appropriate prosody based directly on where it and its partner are in a game.

1.1 Related Work

Prosody in in-game interaction has been studied from several angles. Schafer *et al.* (2005) examined how speakers naturally adjust their prosody during a cooperative game task, finding that prosodic structure consistently reflected syntactic choices regardless of how ambiguous the situation was. This is relevant to our setting because it suggests that prosody in task-driven collaborative environments relates to communicative structure. Vrijders *et al.* (2025) showed that the tone of voice a speaker uses has an immediate and measurable effect on the listener’s willingness to collaborate and their sense of closeness to the speaker, with voice quality — specifically harshness — being a critical factor. Even a single sentence spoken in a controlling versus supportive tone was enough to produce different reactions. This is relevant to our interest in how a guiding player’s prosody might affect the dynamic between participants. Sadoughi *et al.* (2017) found value in an in-game agent’s matching the user’s prosody. Ward and Abu’s (2016) found that specific prosodic constructions are tied to various kinds of events, intents and actions in this game.

In general, prosodic behavior has been modeled in many genres and scenarios, however the relation between position and prosody appears not to have previously been studied.

2 Research Questions

We set out to answer the following questions:

RQ1 How much is the guiding player’s prosody determined by their in-game position?

RQ2 How much is the guiding player’s prosody determined by the other player’s position?

3 Methods

3.1 Data

We used an existing dataset consisting of video recordings and dual-channel audio of two individuals playing the online game *Fireboy and Watergirl* (Ward and Abu, 2016).

In these recordings, one fixed player served as a guide, playing the game in turn with a dozen partners, who were usually unfamiliar with the game mechanics. There were two player roles, Fireboy and Watergirl, and the guiding player always took the Watergirl role, so we below refer to his voice as the Watergirl voice.

The guiding player actively instructed and supported the other throughout gameplay, sometimes interpolating various smalltalk. His speech contains rich prosodic cues associated with instruction, coordination, encouragement, decision-making, and so on. As we have ample data for him, all the analysis below is based on his voice.



Figure 1: Pitch heatmap for Game Number 3, Level 1.

3.2 Spatial Mapping

The first step was to obtain x and y coordinates of Watergirl and Fireboy across every frame of the video. To accomplish this, YOLO (v8) was used, a deep learning model for object detection that works by dividing an image into regions and predicting bounding boxes and class probabilities for objects in a single forward pass. Since the default YOLO training dataset (COCO) does not include the Watergirl or Fireboy sprite images, we created a custom dataset. For this we used the tool Roboflow to manually annotate approximately 100 images by drawing bounding boxes around the two characters. This dataset was used to train a specialized YOLO model for our task. The trained model achieved good detection precision overall, though it occasionally misclassified blue diamonds as Watergirl. Because these false detections were always associated with low confidence values, filtering out all detections with a confidence score lower than 0.60 solved the issue. After this filtering, a reliable stream of x and y coordinates corresponding to the centers of the players' bounding boxes on each frame were obtained.

3.3 Pitch Extraction

The second step was to extract the pitch. Since Watergirl's audio was recorded in the right channel, that channel was isolated and used as input for the REAPER toolkit to calculate the fundamental frequency (F_0) at 5 millisecond intervals. REAPER outputs files containing the timestamp, a voiced/unvoiced flag, and the estimated F_0 value in Hertz. To ensure clean data, unvoiced and silent frames were filtered out.

As accurate pitch extraction is the foundation of everything, we devoted some attention to possible sources of noise and error. We identified the following:

Breathing artifacts. In some instances, the microphone captured the speaker's breathing, and the pitch extraction algorithm produced spurious fundamental frequency (F_0) values for

these segments. Because these values often fell within the normal pitch range, they were difficult to distinguish from valid speech using simple threshold-based filtering. As a result, these artifacts remain a potential source of noise in the dataset.

Low-intensity or breathy speech. When the speaker’s voice became very soft or breathy, the pitch detection algorithm occasionally produced unusually high F_0 values (approximately 300–400 Hz), which are inconsistent with the expected range for the speaker. To prevent distortion of the heatmap results, these outliers were removed through upper-bound filtering of pitch values.

Vocal creakiness. Certain segments of speech exhibited creaky voice. In these regions, the pitch estimation algorithm struggled to produce stable F_0 values, leading to inconsistent or noisy measurements. This variability introduces additional noise in the extracted prosodic features.

Recording artifacts. When the speaker was positioned too close to the microphone, popping sounds were introduced into the audio signal. Although these artifacts were not always perceptible to the human ear, they interfered with the pitch detection process and occasionally resulted in inaccurate F_0 estimates.

We also note that variability between the expertise levels of the participants likely influenced what we obtained. Some players were already familiar with the game, while others were inexperienced and required more guidance. In sessions with experienced players, less communication was needed between participants, meaning the player in the leading role spoke less frequently. As a result, fewer pitch data points were captured in those segments, reducing the amount of prosodic information available for analysis. In contrast, less experienced players required more verbal coordination, leading to increased speech and a richer set of pitch observations. This difference in communication patterns introduces variability in the dataset, as the observed pitch distributions may reflect not only in-game events but also differences in player familiarity and interaction dynamics.

3.4 Heatmap

A heatmap is a way of visualizing data using colors, where different colors represent different values. In this project, it allows us to see how pitch varies across different parts of the game map. Creation of the heatmaps took several steps.

First, the two data sources—YOLO detections and REAPER pitch outputs—were joined. This was accomplished by aligning pitch values and frame positions by their timestamps. The resulting dataset contained, for each time point, Watergirl’s x and y coordinates, frame number, and pitch (F_0 in Hz).

Next the data was separated by level. In the videos, the players went through multiple levels of the game, each consisting of a different map. (Level 1 was present in all videos, while the others varied.) The data was separated by level using the timestamp ranges that we defined in a JSON file.

The game area was then partitioned into a regular grid, where each cell represented a fixed number of pixels determined by the grid-size parameter. In the current implementation, the default grid size is 30 pixels, although this can be adjusted to produce either finer or coarser spatial resolution. Each tracked position was assigned to a grid cell by converting its x and y coordinates into integer bin indices.

Once the coordinates had been aligned, the pitch values associated with all frames falling inside the same grid cell were aggregated by taking their mean. These average values were then stored in a two-dimensional matrix whose dimensions corresponded to the number of

horizontal and vertical bins in the level image. Cells that contained no observations were left empty as missing values, which meant that only visited regions of the map contributed color information.

For each game level, the script loaded a background image representing that portion of the game and then mapped Watergirl’s tracked positions onto that image. The image was taken directly from a frame in the video as to maintain the same pixel resolution that was used when recording the coordinates. Finally, the matrix of average pitch values was displayed as a semi-transparent overlay on the background image for the level. The visualization used a color map to represent relative pitch differences across space, while the transparency allowed the structure of the level itself to remain visible underneath. A color bar was added to indicate the pitch scale in Hertz. (As a side note, before visualization, pitch values were filtered to retain only those within a range of 20 to 200 Hz, so that outliers would not distort the color distribution across heatmaps.)

In this way, the output provided a spatial summary of Watergirl’s average pitch across the areas she occupied during gameplay, as seen on Figure 1.

Additional heatmaps were generated using Fireboy’s spatial positions, but still mapping Watergirl’s pitch values. This modification was motivated by the idea that the guiding player’s vocal behavior may be influenced more by the actions and location of the other player than by their own. Because the guiding participant was already familiar with the game, we speculated that they experienced fewer direct challenges and therefore had less need to express frustration or difficulty related to their own gameplay. Instead, their speech was often directed toward assisting the second player. Thus, analyzing Watergirl’s pitch as a function of Fireboy’s position provided a complementary perspective, capturing how vocal variation may reflect the state and needs of the guided player rather than the speaker’s own.

3.5 Aggregation

To improve the reliability of the generated heatmaps, an aggregation step was introduced following the initial per-session analysis. Data from the same levels across multiple gameplay sessions were combined. Since the layout of each level remain constant across sessions, this allowed positional data from different players and runs to be mapped onto the same map, as seen in Figure 2. By grouping all observations corresponding to the same level, a more comprehensive representation of player behavior and vocal patterns was achieved. This used the same process as the original heatmaps, just with aggregated data.

This aggregation process increases data density within each region, enabling more stable estimates of pitch values. Areas that may have been underrepresented in a single session become more clearly defined when multiple sessions are combined. As a result, the aggregated heatmaps provide a better view of how prosodic features vary. Other selected heatmaps for Watergirl are provided in Appendix A and for Fireboy in Appendix B.

3.6 Smoothing

We also experimented with a variant visualization method. We realized that the strict spatial boundaries introduced by the grid-binning approach were unsuited for the representation of continuous movement. Data points that fell near the edges of bins were treated separately, meaning that useful spatial information about nearby positions could be lost. To address this, an alternative approach based on Kernel Density Estimation (KDE) was explored.

Kernel Density Estimation (KDE) is a technique for turning a set of discrete data points



Figure 2: Aggregated Heatmap for Level 1.

into a smooth, continuous surface. Rather than dividing space into rigid cells like the binning approach does, KDE works by placing a small “bump,” mathematically a Gaussian curve, centered on each data point. These bumps overlap and add together across the entire space, so instead of sharp boundaries between regions, you get a flowing surface where each location reflects the influence of nearby observations. The result is that a data point does not only contribute to its own cell, it also has a fading influence on the surrounding area, with that influence decreasing the further away you get.

How wide or narrow those bumps are is controlled by a parameter called the bandwidth. A small bandwidth makes each bump tight and narrow, so the resulting surface closely follows the individual data points and preserves fine spatial detail, but can look noisy if data is sparse. A large bandwidth makes each bump spread out, producing a smoother surface that is easier to read at a glance, but risks blending together patterns that are actually distinct.

To implement this, a script was developed that computes a Gaussian-kernel-weighted average pitch at each point on a regular spatial grid. For each grid cell, the estimated pitch is calculated as the weighted mean of all observed F_0 values, where the contribution of each observation decreases with its distance from the grid point according to a Gaussian function. This produces a continuous pitch surface, as seen on Figure 3, in Hz units. (As a side note, we also tried a density-weighted approach, but found that this made areas where the players spent more time artificially too prominent.) Coordinates are normalized to the full video frame dimensions before computing distances, so the bandwidth parameter is expressed in that normalized space, where a value of 0.01 corresponds to roughly that fraction of the frame width.

Despite producing smoother heatmaps, this script was not retained as the primary visualization method. Because the goal was to directly represent average pitch across space rather than a continuous interpolated surface, the binning approach seemed more interpretable. However, as described in Section 5, a modified version of the Gaussian KDE approach was



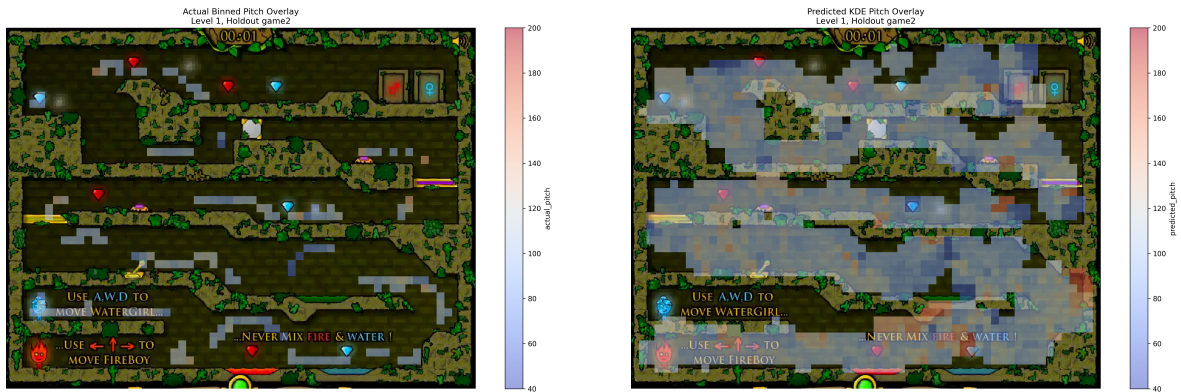
Figure 3: Smoothed heatmap for Game Number 2, Level 1.

adopted for model evaluation, where its ability to generalize across small positional gaps made it better suited for cross-session comparison.

4 Observations

Examining Level 1, there seems to be a tendency for the pitch to be higher as Watergirl is approaching obstacles, and lower after she's past them. However this was not seen for the other levels. Overall, the visualizations did not provide the expected insights.

5 Predictive Models



(a) Heatmap for game Number 2, Level 1.

(b) KDE Predicted Heatmap for Level 1.

Figure 4

After generating the heatmaps, we wanted to determine whether the observed patterns reflected meaningful and consistent prosodic behavior or were simply results of individual gameplay sessions. To address this, a validation step was introduced using a leave-one-out prediction/evaluation framework. The goal of this approach was to assess whether spatial patterns in pitch actually existed and to answer our research questions.

We utilized the aggregated dataset from the heatmap generation. Then, for each iteration, one gameplay session was excluded and the remaining sessions were used to construct an average pitch map. This aggregated map was used as a prediction of expected pitch behavior, while the excluded session served as the ground truth. By comparing the two, it was possible to evaluate how well the aggregated data captured patterns present in gameplay. We only used levels that were present in at least 3 gameplay sessions, to ensure we had enough data for a valid result. There were, accordingly, only 8 usable levels.

The initial implementation of this evaluation followed the same grid-based binning approach used in heatmap generation. Each level was divided into fixed spatial bins, and the average pitch within each bin was computed for both the aggregated dataset and the held-out session. Corresponding bins between the two maps were then aligned, and the Pearson correlation coefficient was calculated using only bins that contained valid data in both maps. This provided a quantitative measure of similarity between the predicted and actual pitch.

While this method offered a straightforward way to compare maps, it demonstrated the same limitations with binning that were described earlier. The use of strict spatial boundaries meant that neighboring regions were treated as independent, preventing the model from capturing continuity in player movement and prosodic behavior. Small positional differences between sessions could affect different bins, even when underlying patterns were similar, reducing the sensitivity of the comparison.

To address these limitations, an alternative evaluation using Kernel Density Estimation (KDE) was explored. As introduced in Section 3.6, KDE produces a continuous pitch surface by computing a Gaussian-kernel-weighted average of observed F_0 values at each grid point. In the evaluation context, this approach was applied with one important difference: rather than computing the KDE directly from raw positional observations, average pitch values were first computed within spatial bins, and the Gaussian kernel was then applied to spread the influence of these aggregated values across neighboring regions.

In other words, KDE-based evaluation in this stage differed slightly from the KDE approach used in the previous section. Rather than applying KDE directly to individual data points, this implementation operated on the binned representation of the data. This allowed for a smoother and more continuous comparison while still preserving the structure of the grid-based approach. By contrast, the earlier KDE method was applied directly to raw positional data, where each individual observation contributed independently to the density estimate. This hybrid approach provided a more comprehensive spatial trajectory by incorporating both aggregation and smoothing, helping to reduce the impact of noise and small positional variations.

We computed the average correlation per level for both Watergirl and Fireboy.

6 Results

As seen in Table 1, when averaging the correlations across all analyzed levels, the smoothed method did better overall, and showed more value for the Watergirl features, which one would expect.

Player	Binned-based Predictions	KDE-based Predictions
Speaker (Watergirl)	-0.039	0.067
Other Player (Fireboy)	0.061	0.020

Table 1: Average of per-level correlations between actual and predicted pitch values, across all analyzed levels.

As a side note, it is possible that the effectiveness of smoothing depends on the role and movement behavior of the player. Perhaps the player controlling Fireboy typically follows a more structured path, and such movement patterns may be better preserved through direct spatial binning, while Watergirl’s broader movement and guiding behavior may benefit more from the smoothing introduced by KDE.

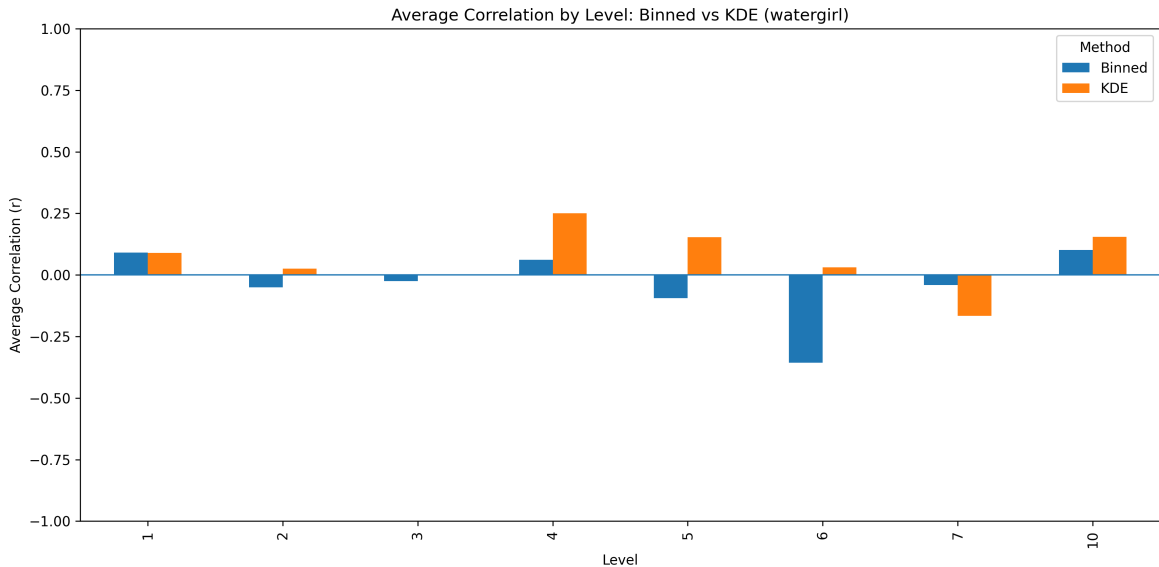


Figure 5: Watergirl correlation results

The performance per level varied greatly. As seen in Figure 5, for Watergirl, the highest correlation was about 0.25 using the KDE-based approach. As seen in Figure 6 the highest correlation for Fireboy occurred in Level 5, with an average value of approximately 0.41 using the binned method. However, for most levels the correlation values were low often near zero or even negative 6. This suggests that the relationship between spatial position and pitch is weak or inconsistent across gameplay sessions.

Additionally, the reliability of these results is limited by variability in the dataset. For example, the highest Fireboy correlation (Level 5) was computed from only three gameplay sessions, so we can’t take it as reliable. In contrast, Level 1, which had the most data and appeared in all 15 gameplay sessions, consistently produced low correlation values. This indicates that with a larger sample size, strong spatial patterns in pitch were not observed.

Overall, these findings indicate that spatial position alone is not a strong predictor of pitch variation in this dataset. While certain levels show moderate correlations, and while we could probably improve the performance somewhat by tuning the smoothing parameter or by better modeling choices, the lack of consistent results across levels does not support the idea

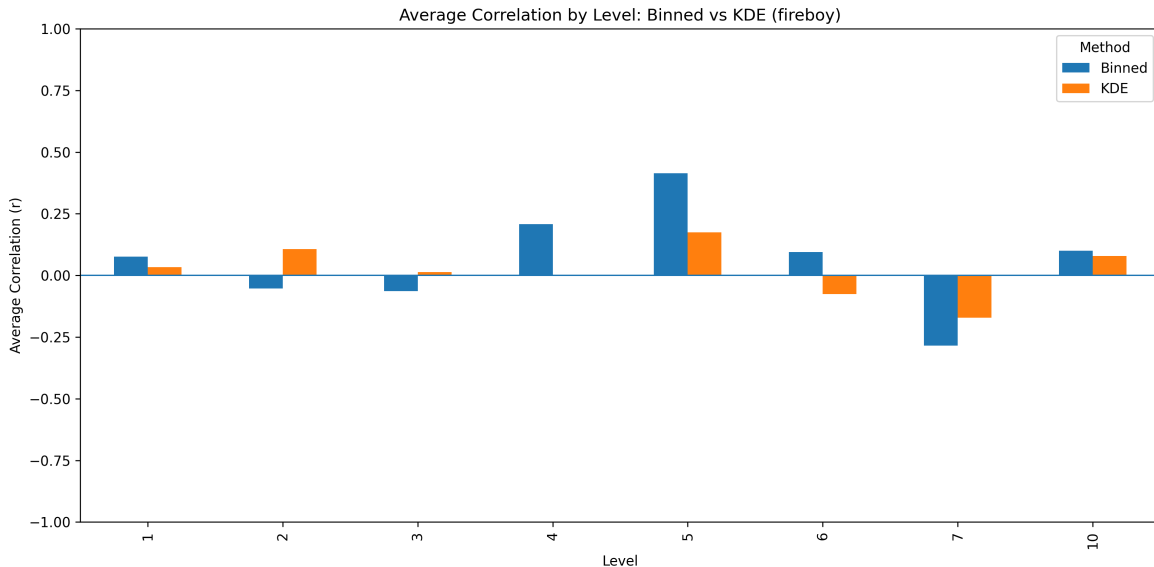


Figure 6: Fireboy correlation results

that we can often find a stable location-prosody relationship.

7 Conclusion

The evaluation results did not reveal strong or consistent correlations between spatial position and pitch.

This highlights the complexity of prosodic behavior and suggest that there are multiple factors that influence vocal expression during gameplay, for example, variability in player experience, communication dynamics, microphone artifacts, and differences in interaction styles.

Another outcome of this work is the code. The code is available upon request. While we only used it to investigate pitch, the methods and scripts created can be readily extended to other prosodic features such as intensity, speech rate, voice quality or phonetic reduction. This code could also be adapted for analyzing and visualizing prosodic features in other games, and in human-human or human-robot situated action in real environments.

References

- [1] Nigel G. Ward and Saiful Abu. Action-Coordinating Prosody. *Speech Prosody*, 2016.
- [2] Nigel G. Ward, Fernando Alvarado, Harm Lameris. How Much Value can Appropriate Prosody have for a Collaborative Agent? manuscript, 2026.
- [3] Najmeh Sadoughi, André Pereira, Rishub Jain, Iolanda Leite and Jill Fain Lehman. Creating prosodic synchrony for a robot co-player in a speech-controlled game for children. *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, pp 91-99, 2017.

- [4] Berdien Vrijders, Netta Weinstein, Silke Paulmann, Bart Soenens, Joachim Waterschoot, and Maarten Vansteenkiste. Your Prosody Matters: The Effect of Controlling Tone of Voice on Listeners' Experienced Pressure, Closeness, and Intention to Collaborate With the Speaker. *Motivation Science*, 11(1), 49–66, 2025.
- [5] Amy J. Schafer, Shari R. Speer, and Paul Warren. Prosodic Influences on the Production and Comprehension of Syntactic Ambiguity in a Game-Based Conversation Task. In A. Cutler (Ed.), *Twenty-First Century Psycholinguistics: Four Cornerstones*. Lawrence Erlbaum Associates, 2005.

A Watergirl Aggregated Heatmaps



Figure 7: Aggregated Heatmap for Level 1.



Figure 8: Aggregated Heatmap for Level 2.



Figure 9: Aggregated Heatmap for level 3.

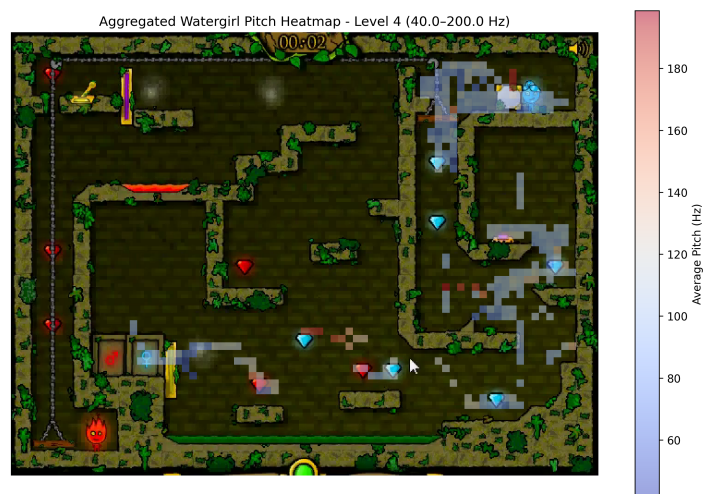


Figure 10: Aggregated Heatmap for level 4.



Figure 11: Aggregated Heatmap for level 5.

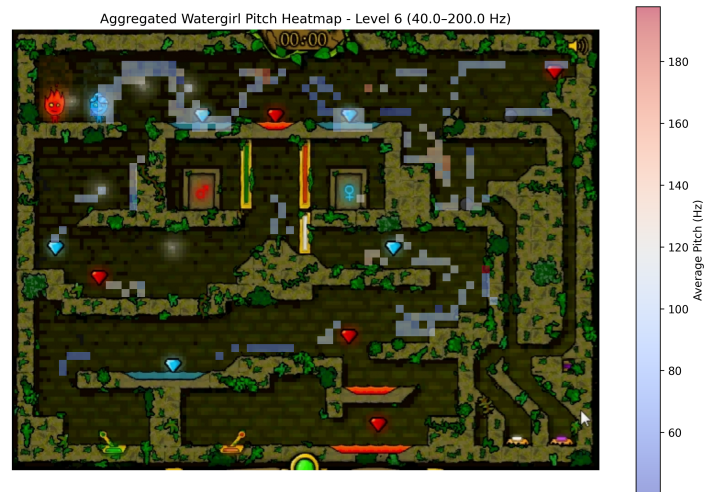


Figure 12: Aggregated Heatmap for level 6.

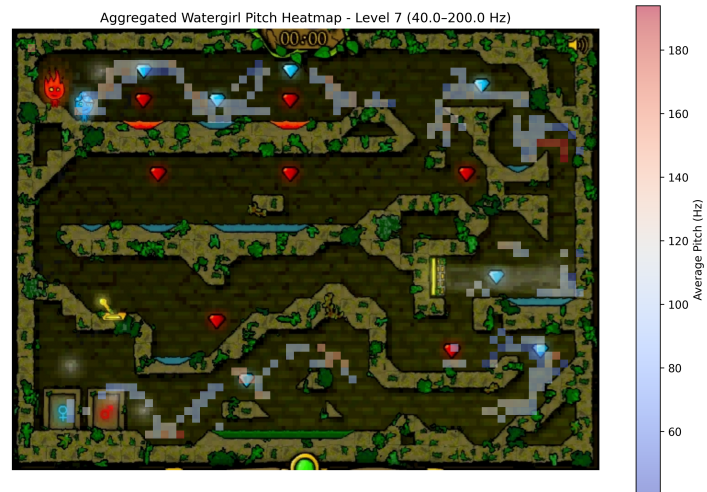


Figure 13: Aggregated Heatmap for level 7.



Figure 14: Aggregated Heatmap for level 10.



Figure 15: Aggregated Heatmap for level 11.

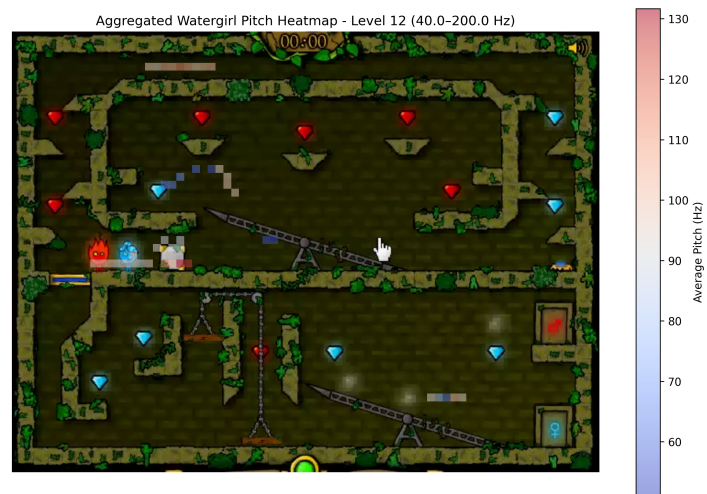


Figure 16: Aggregated Heatmap for level 12.



Figure 17: Aggregated Heatmap for level 14.

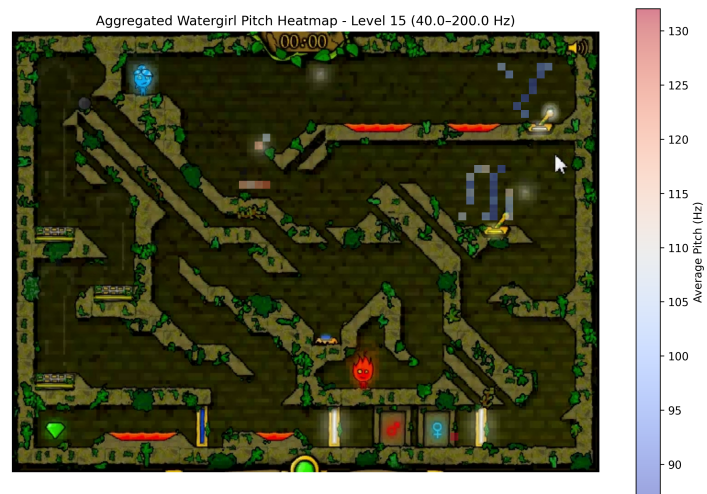


Figure 18: Aggregated Heatmap for level 15.



Figure 19: Aggregated Heatmap for level 19.



Figure 20: Aggregated Heatmap for level 20.



Figure 21: Aggregated Heatmap for level 22.



Figure 22: Aggregated Heatmap for level 23.



Figure 23: Aggregated Heatmap for level 24.



Figure 24: Aggregated Heatmap for level 25.



Figure 25: Aggregated Heatmap for level 26.



Figure 26: Aggregated Heatmap for level 28.

B Fireboy Aggregated Heatmaps



Figure 27: Aggregated Heatmap for level 1.

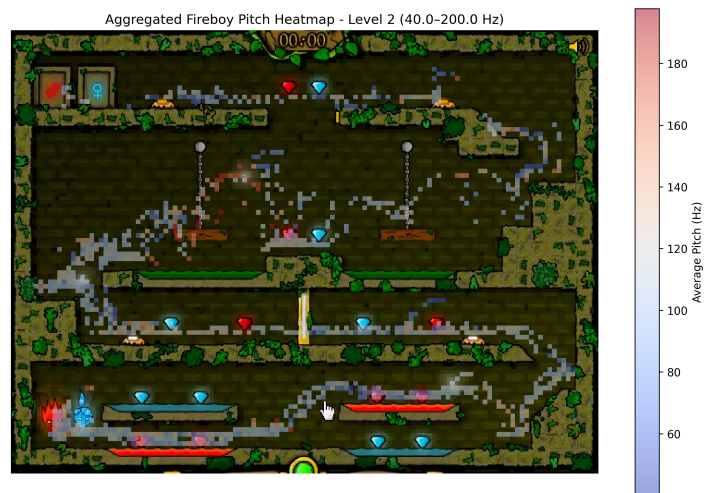


Figure 28: Aggregated Heatmap for level 2.

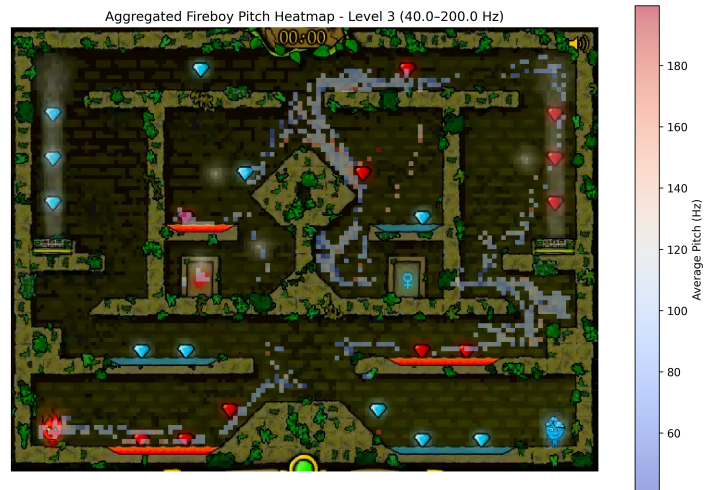


Figure 29: Aggregated Heatmap for level 3.



Figure 30: Aggregated Heatmap for level 4.



Figure 31: Aggregated Heatmap for level 5.

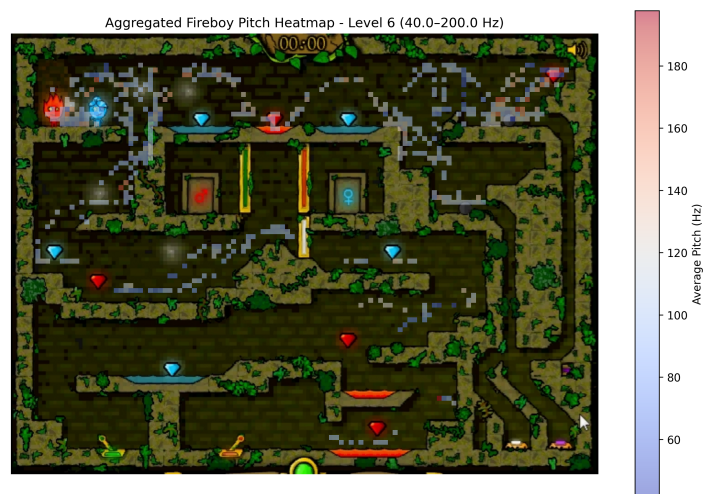


Figure 32: Aggregated Heatmap for level 6.



Figure 33: Aggregated Heatmap for level 7.



Figure 34: Aggregated Heatmap for level 10.



Figure 35: Aggregated Heatmap for level 19.



Figure 36: Aggregated Heatmap for level 25.

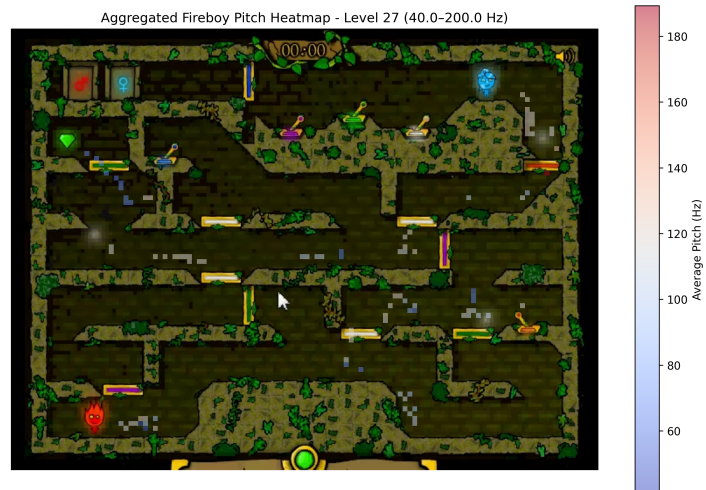


Figure 38: Aggregated Heatmap for level 27.



Figure 37: Aggregated Heatmap for level 26.



Figure 39: Aggregated Heatmap for level 28.