Foosblitz AI Commentator

Bachelors TCS Design Report

Authors
S. Csapó
D. Hernández García
E. Malafronte
P.J. Meijer
N. Thomma

Supervised by
dr. L. Gatti
M.C. Pijnappel MSc

Faculty of EEMCS
University of Twente
April 17, 2025

Abstract

This report describes the design and implementation process of an AI commentator for the Foosblitz smart foosball table at the University of Twente. The goal is to enhance the experience of foosball matches by providing real-time, engaging commentary. Especially for spectators, who often miss the excitement and context without commentating. We combine data from the table with a Large Language Model and a Text-To-Speech system, so that it can interpret live game events and generate exciting commentary in real time. The process involved careful planning, design, and testing with users to make sure the system was responsive and fun.

Contents

1 Introduction **3**

2 Method Approach **5**
2.1 Planning 5
2.2 Organisation 5
2.3 Requirements 6

3 Requirements **7**
3.1 Must 7
3.2 Should 7
3.3 Could 8
3.4 Won't 8

4 Research **9**
4.1 Foosblitz 9
4.2 Foosball 10
4.3 Open Source Football Videogame AI Commentator 12
4.3.1 TTS 12
4.3.2 LLM 12

5 Design **13**
5.1 System flow 13
5.2 Threading 15
5.3 Foosball rules 15

6 Implementation **16**
6.1 Foosblitz 16
6.1.1 The code 16
6.1.2 The table 17
6.2 AI Commentator 17
6.2.1 Large Language Model (Claude, Gemini) 17
6.2.2 Text-To-Speech (Coqui TTS) 18
6.2.3 Detection Model (Roboflow) 18

7 Testing **20**
7.1 Test Plan . . . 20
7.1.1 Integration Testing . . . 20
7.1.2 User Testing . . . 20

8 Evaluation **21**
8.1 Performance . . . 21
8.1.1 Must Have . . . 21
8.1.2 Should Have . . . 22
8.1.3 Could Have . . . 22
8.2 Possible Improvements . . . 23

9 Discussion and Conclusion **24**
9.1 Planning . . . 24
9.2 Task Division . . . 25
9.3 Team Evaluation . . . 25
9.4 Final Results . . . 26
9.5 Improvement Points . . . 26

10 Future Work **27**

Appendices **29**

A Prompts **30**
A.1 System Prompt . . . 30
A.2 Event Prompt . . . 32
A.3 Filler Prompt . . . 32

Chapter 1

Introduction

For this design project, the group has made an AI commentator to pair with a smart foosball table. Within the definition of our project, a smart foosball table is a table with an automatic goal-counter, with LED lights lighting up red and blue, and a web application showing camera feed from above the foosball table, the score, and information on the game. Our AI commentator is intended to enhance the experience of an exciting foosball match. Imagine a foosball match without a commentator, as a player you might enjoy this since you are invested with a goal in mind. However, as a spectator it would not be as exciting, since a game without a commentator is unpassionate, muted, and uninspiring. For spectators, it might even be confusing without the guiding word of a commentator, losing focus of the ball or what is happening in the game. A commentator thus brings a lot to the table and is very important for the ambiance. For a casual foosball match, a commentator is not always available. This is where an AI commentator would fill the market gap. An AI commentator is readily available for anyone with a foosball table and the correct installation. This way a lively atmosphere can be brought to any fun match.

This project was designed for an already existing smart foosball table called Foosblitz [8], located in the IoT Cyberlab at the University of Twente, where our clients are based. The foosball table is used by students and staff to get a break from their educational pressure, by relieving stress and relaxing with a fun game of foosball. Our goal is to enhance the Foosblitz by creating an Artificial Intelligence (AI) commentator for the table. We propose sending the live data we receive from the table (such as goals, last kicker, and possession) in a predefined prompt template to a Large Language Model (LLM), like Gemini[1] and ChatGPT[2] that can understand and generate human language. The LLM will be responsible for turning the live data into lines of commentary. Next, this commentary will be sent to a Text-to-Speech (TTS) model, making an audible commentary out of the received text from the LLM. This way we can achieve close to real-time commentary with the use of Artificial Intelligence.

[1] http://gemini.google.com/
[2] https://chatgpt.com/

To avoid confusion, throughout this report we use the term foosball player to denote the red and blue plastic foosball players inside the table, and human players to denote the humans playing the game.

The report first details our initial proposal to our clients, followed by the requirements we constructed and the research we carried out on the TTS and LLM. Subsequently, the design, implementation, and testing phases of our project are explained, ending with the evaluation of our results, our discussion on our team dynamics and possible future work to be made.

Chapter 2

Method Approach

In this section, our proposal is explained. This proposal has been used to communicate with the supervisors about the product envisioned to be built. It contains preliminary requirements, which allow room for adjustment since the design process and implementation might be different than expected. Next, it also contains envisioned phases and the project's planning. Lastly, the proposal discusses project organization. This proposal has been discussed with the supervisors before continuing to build the actual product.

2.1 Planning

Next, planning was discussed. This had the purpose of showing what the supervisors can expect at what times, and the purpose for the project members to see whether the planning is realistic and doable. Figure 2.1 was made to visually show our planning. This Gantt chart is reflected upon with greater detail in Section 9.1.

2.2 Organisation

Some agreements were set on how the group would organize their work. In order to ensure the team is on the right track, weekly meetings will be held with the supervisors. In these meetings, an update will be given by the team on what they are currently keeping busy with and what the supervisors feel is important to share, this meeting is also an opportunity to ask questions, ask for help, or request input.

Within the team, tasks will be divided and responsibilities will remain shared. We will split the team for example into two, to work on the LLM and TTS simultaneously. When questions or issues arise, the team will all look together to overcome these.

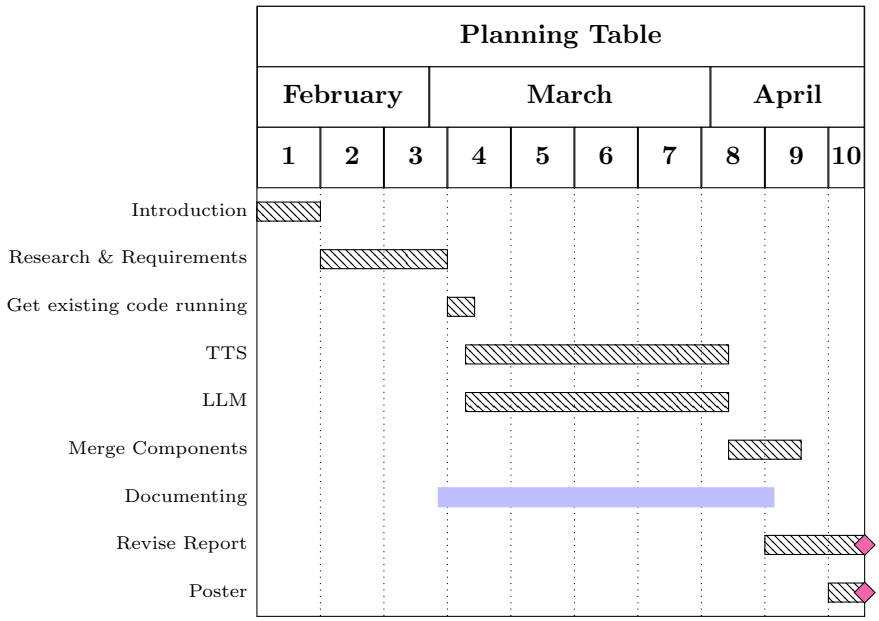

Figure 2.1: Planning

2.3 Requirements

To get a better understanding of what the AI Commentator should be able to do, during supervisor meetings questions were asked to grasp what our supervisors had in mind. We talked to our clients and stakeholders to create a final requirements list, detailed in the following section.

Chapter 3

Requirements

For the formulation of the requirements, the MoSCoW method was used. This method uses **M**ust haves, **S**hould haves, **C**ould haves, and **W**on't haves to distinguish the different importance of each requirement. These are all the requirements we ended up implementing and the ones we did not end up implementing. In Chapter 6 we describe our implementation process and which requirements we implemented, which we decided not to implement with our reasoning behind these decisions.

3.1 Must

- Have a delay of less than 2 seconds between an event (e.g. a goal) and commentary about this event.
- Have the commentator speak understandable English.
- Have the commentator speak with a thrill and intonation.
- Have some commentary at least every 2 seconds, it should not be silent for more than 2 seconds.
- Commentate on goals, possession, overtake/ball switches teams, serves, and out-of-plays correctly.

3.2 Should

- Have one of our/staff voices for the commentator.
- Recognize more moves to commentate on.
- Recognize higher level techniques to commentate on.
- Commentate on assists, dead balls, and faults correctly.

3.3 Could

- Have an actual commentator's voice.
- Recognize situations that are almost happening to comment on.
- Commentate on the previous commentary whenever something similar happens. This could be useful as filler commentary if the 2-second silence is approaching.
- Provide users the option to select a team on the web application, for example, the blue team could choose to play with Manchester United players.
- Have current commentating halt in case a more important event happened; if it is commentating on a pass, have it halt to commentate on a goal.

3.4 Won't

- Use a camera to look at the human players and use that for commentary.

Chapter 4

Research

Before we could begin designing our project, we had to research the different methods we could use to implement our requirements. This research is described below.

4.1 Foosblitz

Figure 4.1: Foosblitz table.

As mentioned before in our introduction, our AI commentator is an expansion on the Foosblitz smart foosball table, built by a previous Bachelor TCS Design group [8]. To get an understanding of what the table is designed to do and what our basis is for our project, we investigated their design report and interacted with the physical table. The smart foosball table is equipped with a camera at the top and LEDs around the frame. A picture of the smart foosball table is shown in Figure 4.1. The camera has a view of the complete foosball field, which is useful to keep track of the ball and the foosball players. The LEDs turn white to assist the camera with calibrating the corners of the table; they blink blue when the blue team scores a goal and vice versa for red. The LEDs at the bottom of the frame count the score of each team. When the red team has scored three points, the bottom three LEDs are green, and vice versa for blue, with pink LEDs for the points.

There is a separate scoreboard that shows the current score of the game. It is automatically updated when a goal is scored or when the game is reset. The game can be reset by pushing a small

button behind either of the goals. There is also a webapp that shows the current score, the live camera feed, a delayed camera feed, the average frames per second, the name of the last kicker (the plastic player who makes the goal), and buttons for calibrating the camera and seeing the camera feed on full screen.

According to the report, in the backend of the foosball table, certain factors are kept track of, such as the ball's location, ball possession, duration of the possession, and which foosball player kicked the ball last.

To detect the ball position, the Foosblitz group used an object detection model, made in roboflow [3]. Their specific model version is called Roboflow 3.0 Object Detection, which is derived from YOLOv8 [1]. YOLO is an object detection model, which approaches detection like a regression model [7].

To detect the players, they use colour masks, which filter an image for pixels within a certain colour range. Specifically, the HSV colour spectrum Figure 4.2 was used to achieve this, this spectrum stands for Hue, Saturation, and Value. This spectrum is the best choice for colour selection, as it gives you a lot of control to select colours of a specific hue.

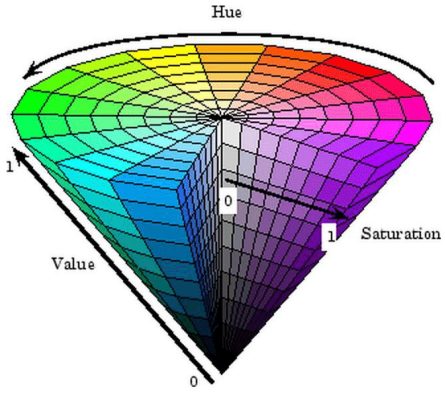

Figure 4.2: HSV Colour Range

4.2 Foosball

Now that we have looked at the technical aspects of the foosball table, we need to consider the foosball game and the rules. An image of a foosball table is shown in Figure 4.3. Each metal bar with foosball players attached to it is called a rod. There is for each team, a goalkeeper rod, a defender's rod, a midfield rod, and an attacker's rod. Most people know the foosball game, and a lot even have played a foosball game once. But, how many people actually know the rules? Some people might know some basic rules, such as you are not allowed to do an uncontrolled spin of a bar. But there actually are more rules to a foosball game than one might expect. The 6 basic rules [2] of foosball are stated as :

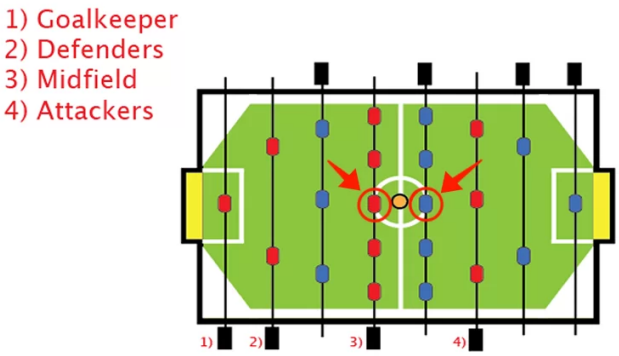

Figure 4.3: Top Down View of Foosball Table Diagram [6]

1. A coin flip decides which team may start with the first serve.
2. You are not allowed to spin a rod.
3. You are not allowed to jar a bar to distract the opponent or to make the opponent lose possession of the ball.
4. A dead ball should be reserved by the team that served last. After a goal, the team that was scored on can serve.
5. If a ball leaves the table, the ball should be reserved.
6. The ball may not be in the possession of one bar for more than 15 seconds at any time.

These are not all rules, a guide from Imperial [4] describes more rules and game events, it describes a goal as "A goal is when the ball passes through the goal hole. If your goalie hits the ball into your goal, it is considered an own goal. Even if the ball bounces back out, it is still considered a goal. However, if the ball is served and goes into the goal without anyone touching it, official foosball rules state that it is not a goal."

The smart foosball table, Foosblitz, is currently located in the IoT Cyberlab, at the University of Twente. It is most used by the PhD students who have their office space on that floor. We have discussed with them what rules they would like implemented, as this might improve their experience. They've mentioned they would like the following rules implemented:

- It is a foul when an attacker passes the ball to another attacker on the same rod, and then scores.
- It is a foul when an attacker passes the ball to himself and then scores.

These are allowed when the ball touches an opposing player, a player on a different rod, or the wall.

4.3 Open Source Football Videogame AI Commentator

We have gained a substantial part of our inspiration for the LLM and the TTS part of our project from an Open Source Football Videogame AI Commentator, which was provided to us by our supervisor. This was originally a research project done by Michał Czaplicki [5], but the version we received was improved by dr. Lorenzo Gatti. The TTS has been custom-trained, and the LLM has been changed from GPT-3 to Claude[1].

4.3.1 TTS

We have found that in the open source football game system, the Coqui TTS library is used, which influenced our decision on which TTS to use. We subsequently decided to only seriously look at the Coqui TTS library as it was optimal for us since it allows us to train our own models from scratch. In the Open Source Football Videogame AI Commentator, a trained model, with a commentator voice, was already used. This was trained with the Variational Inference with adversarial learning for end-to-end Text-to-Speech (VITS) system. In the first version of our code, we opted to use the same model.

We did research on different TTS systems, but most of them had one crucial problem regarding our project. We needed a model that is possible to train with our own data, so a unique commentator voice can be achieved. This was not possible in most systems. One that we were interested in trying out was Kokoro, as it was said to be the most realistic voice out of all the open source TTS systems. However, just as we found with other ones, also with Kokoro we could not train our own model; we could only mix a few predefined voices, which was not sufficient for our needs.

4.3.2 LLM

We did not have to research many LLM systems as Claude was already implemented in the Football Videogame AI commentator system. Claude was also an optimal choice for us, because it is fast and reliable, and the documentation on it is extensive. It made our job significantly easier to see the connection of a TTS and LLM already made in the code of a working AI commentator system. This meant we could get inspiration on how to separate the two components and how to run the threads the LLM and TTS should be on.

[1] https://claude.ai/

Chapter 5

Design

In this section we describe the design of our system and justify our design choices. We begin with the design of the system with diagrams showing how the information flows, and follow with our design on what foosball rules we will implement.

5.1 System flow

Firstly we had to think about the flow of information in the system - what part of the system knows what, which parts does it communicate with, and what information should it communicate to what parts? We have made a rough sketch in Figure 5.1 that shows how the system would flow. First, an event would happen (detected by our model explained in Section 6.2.3). This could be a goal, a foul, or a 3-second silence. In all cases, a prompt would be sent to the LLM, either a goal prompt, foul prompt, or filler prompt. The Large Language Model (LLM) would take the prompt and generate commentary. This commentary would then be sent to the Text-To-Speech (TTS) who would speak the commentary out loud.

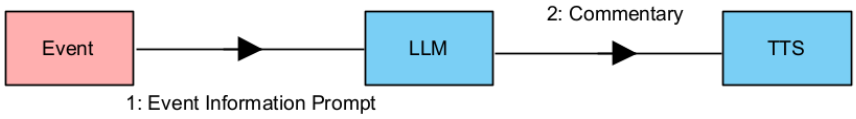

Figure 5.1: System Flow

In Figure 5.2 we made an activity diagram that shows how the system interacts with the LLM and TTS in more detail. Upon startup of the system, the camera needs to be calibrated, so that the detection functions as it should. This cali-

bration is already implemented in the Foosblitz. After this, a system prompt[1] would be sent to the LLM, informing the LLM to generate commentary for a live foosball game, that commentary messages can be of a maximum length of 8 words, and that it should be exciting and interesting. The prompt also tells the LLM in what format the future prompts will be and what kind of information they will contain. This also generates a welcome message, welcoming spectators to the game and kicking off the excitement.

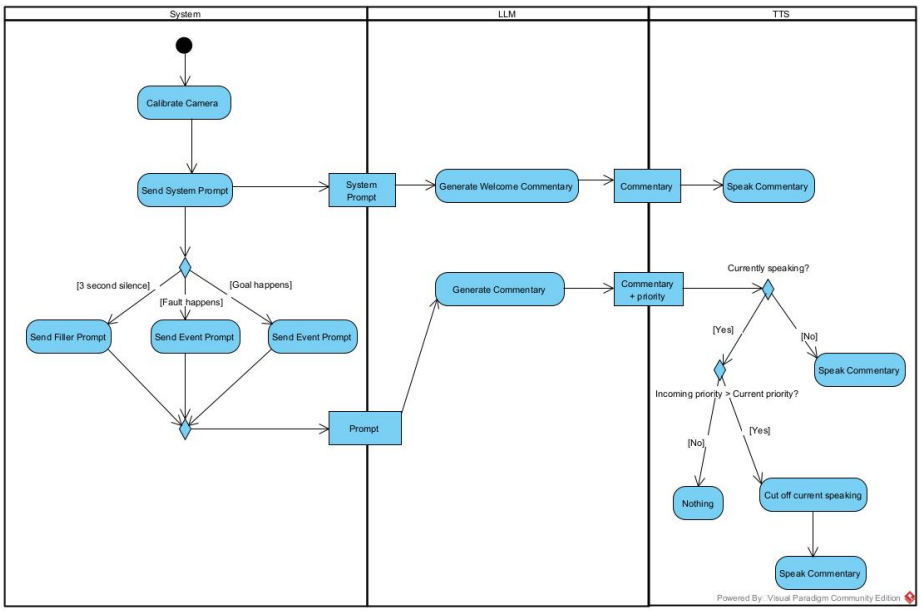

Figure 5.2: Activity Diagram

After initialization, one of three things need to happen: either a goal happens, a foul happens, or nothing happens and it has been silent for 3 seconds. Any of these three will instantiate a prompt. This prompt is sent to the LLM containing information on what happened, which will result in commentary. Once this commentary is sent to the TTS, one of two things can be happening. Either the TTS is currently speaking, or it is not. If the TTS is currently speaking, it needs to decide whether to cut off current commentary to speak the new incoming commentary, or ignore the new incoming commentary and continue the current commentary. This choice depends on the priority of the event that happened. A goal has the highest priority, then welcoming commentary or serves, then fouls, and finally, filler commentary has the lowest priority.

To make the cutting off of commentary possible, we need to design how to do this with threads.

[1] The system prompt can be seen in Section A.1.

5.2 Threading

We decided to run the system in 3 threads. The first one is the thread of the system itself. This handles everything that happens on the Foosball table (goals added, possession counting, ball detection, player detection, et cetera). We need this to be on a different thread than the commentator since we still need the game to run and collect data while the TTS speaks.

The second one is the thread that keeps track of the time that has passed in the game without commentary. When it reaches 3 seconds and no new commentary has started, it sends a filler prompt to the LLM.

The third thread will receive commentary responses from the LLM and send this to the TTS. This thread is needed so that the TTS cuts itself off only at appropriate times, when the priority of the message is higher than that of the one being spoken.

5.3 Foosball rules

In our research Section 4.2 we found that foosball has various rules, which might not all be known to recreative players, who mainly play for fun. Anecdotally, most of these rules were not known to the group members before our research. Since a foosball match does not have an official referee present, players might ignore some rules for the ease and fun of casual play. Therefore we need to think about which rules we want to commentate on, to not confuse or hinder the casual game.

The rules we have designed to implement are:

- A team scores a goal by kicking the ball into the goal of the opposing team.
- When one team scores, the opposing team can serve.
- It is a foul is the ball is in possession of one zone for more than 10 seconds. A zone is one attacking rod, or the two defending rods together.
- It is a foul when an attacker passes the ball to another attacker on the same rod, and then scores.
- It is a foul when an attacker passes the ball to himself and then scores.

These rules offer guidance to human players and spectators without overflowing them with confusion. We also intend to mention the foul with the commentator but to not assume the game is over or assume any other consequences. This will allow people to choose how strict of a game they would like to play without interference.

Chapter 6

Implementation

In this chapter, we detail how each component of our project was implemented. It starts by explaining how we have implemented the existing Foosblitz code and continues on how we have implemented our original AI commentator code.

6.1 Foosblitz

We were given the Foosblitz git repository so we could clone the code and have the Foosblitz project as our basis. In the git repository, a README file was given with instructions on how to install and run the project. This, however, did not go as smoothly as we expected.

6.1.1 The code

With regards to the code, we decided to continue coding in Python in order to keep extending on the already-existing code from the previous design group. We also decided to create a repository in GitLab[1] with our student accounts from the university to be able to work on the code and update it accordingly over time. Most of our implementation can be found in the AICommentator.py file, where we coded how and when to send instructions to the LLM as well as connecting to the TTS to speak out the produced commentary by the LLM. However, we also modified existing files like game.py and detection.py for the purpose of successfully connecting the AICommentator.py file with the game, and also to improve the player detection with the new detection model.

More specifically, the most important code files to take into account for this project are: run_system.py, AICommentator.py, dectection.py and game.py. The main file that runs the code is run_system.py, which runs the already-existing website, which in turn also runs an instance of the Foosball game (game.py) and initializes the camera. The game.py file, as the name indicates,

[1] https://gitlab.utwente.nl/

runs the Foosball game for the table and keeps all the stats stored. Other than this, this file also runs the AICommentator.py file and is connected to the detection.py, which is in charge of the detection of the ball, the players, calibrating the camera,a and determining the different zones in the field.

6.1.2 The table

As previously explained, we were provided with the Foosball table that was made by the previous design project group. This table is equipped with an Arduino stuck to the side of each goal that serves as a sensor to detect when a goal has been scored. Aside from this, a thin wooden cuboid frame with LED strips was also mounted on top of the table by the previous team for decoration purposes. However, the LEDs were set to display white light when calibrating the camera, which is placed in the center of the top wooden frame. These white lights made calibration find all four corners and thus succeed a lot faster. Moreover, when a goal is scored, the LEDs would start blinking to indicate this event. We were forced to disable the LED strip directly above the table since this confused the object detection into thinking that the reflection of an LED on the table might be the ball. As a final observation, the camera has an outgoing cable which we had to connect to the laptop running the code for the Foosball table in order to retrieve all the data from a game, as well as display the footage on the given website for the users.

6.2 AI Commentator

In this section, we will describe our planned and actual experiences with the platforms we are using, and explain which Large Language Model and which Text-To-Speech system we implemented in our project and how.

6.2.1 Large Language Model (Claude, Gemini)

It is essential for a real-time AI commentator that the Large Language Model on which it is based has a fast response rate. Furthermore, it also has to adapt well to performing as a foosball commentator the whole time it is running. We chose to work with Claude, as we have already discussed in Section 4.3.2, with the model "claude-3-5-haiku-20241022" via API.

Throughout most of our work, we did not have any problems with Claude. The response rate was between 0.5 and 1 second in most cases, and the responses were precise. We did have to adapt our prompts several times because the filler commentary lines in the beginning were repetitive, but that was because our prompts were not specific enough. The final filler prompts we used are detailed in Section A.3. We used a system prompt at the beginning of the game in order to give all the instructions to Claude, found in Section A.1
.These include the following:

- Guidelines for commentary

- The format of the prompts we will send
- How the data from the table will be provided in a prompt
- The different types of commentaries needed (Intro, Action, Filler)

Afterwards, we sent messages of the same format, including the event description of what has happened (see Section A.2), the state of the match, and the type of commentary. With this information, we received precise and realistic commentary lines from Claude.

We carried this approach into the last two weeks of our project, but in the end, the servers of Anthropic got overloaded too much, and our product became less and less enjoyable. For this reason, we switched to Gemini, which has a few differences compared to Claude. The first difference we saw was that it was faster, probably because the servers were larger or more responsive. But the most important difference we discovered was that if the server was overloaded, Gemini would not wait 20 seconds to generate the commentary, but would instead send an error message. This helped us a lot, since now we could catch the error and instantly return a commentary line of our choosing, instead of waiting several seconds and possibly breaking our code in case of an overload. This fix was the last step in implementing the LLM in our system.

6.2.2 Text-To-Speech (Coqui TTS)

The Text-To-Speech model of the AI commentator must have an enthusiastic and realistic commentator voice that can pronounce English words and also be seemingly instant in converting our commentary lines to audio. These were the most important deciding factors when we chose a TTS model for our project. We decided to use the models that we were already provided by the Open-source Football Videogame commentator code. This was a model trained in Coqui TTS. The implementation did not cause many problems. The only difficult thing we faced about generating and speaking of commentary lines was the threads they were running on. With the threads, we aimed to achieve that different actions had different priorities. This way, if a goal is scored, for example, it would cut off the previous commentary, so the players and the audience could hear a more accurate and essential commentary of the match. We managed to achieve this by running the TTS on a different thread, which checked for any commentary still speaking and for its priorities.

Later in our project, our supervisor successfully trained a new model in Coqui TTS library, using VITS, the same way as discussed in Section 4.3.1, which he then provided for us. This model ensured a more realistic voice and a more enthusiastic intonation. We have this model in our final product.

6.2.3 Detection Model (Roboflow)

For the detection of the players, the previous project group had chosen to detect them using color masks, which work by filtering the image for colors in a

specific range. This worked suboptimally because the LED lights around the table showed in the reflective metal bars, leading to being detected as players.

For this reason, to improve player detection, we have chosen to use an AI model created in Roboflow. Roboflow is a website that allows users to annotate their own image sets and create all kinds of computer vision models. The previous group has used this website to create a model for detecting the ball [3].

To create a model for detecting the players, we reasoned that we did not need a lot of images to annotate, because of the limited positions each player can occupy. At first, we decided to test the quality of the model with 40 images annotated. This already gave a pretty good model, which worked great for our purposes, so we decided to increase the model to 60 annotated images to increase the confidence and robustness of the model. An example annotated picture can be seen in Figure 6.1.

After annotating the pictures, they were preprocessed into 144 images by first splitting the 60 images into 42 meant for training, 12 meant for validating, and 6 meant for testing. The 42 training images were then copied and augmented twice, all by randomly zooming in between 0 and 20 percent and by rotating somewhere between 5% and 5%. This augmentation is done to make the model more rigorous in case someone accidentally bumps the table.

Finally, after the augmentation the images we started to train a YOLOv12 model on these images. This model was mainly chosen because the previous group had also made a YOLO model as explained in Section 4.1 [8], and we wanted to use the same resources as them if possible. This model ended up working really well with about a 90% accuracy, which is not probable, as we just need the general position of the players.

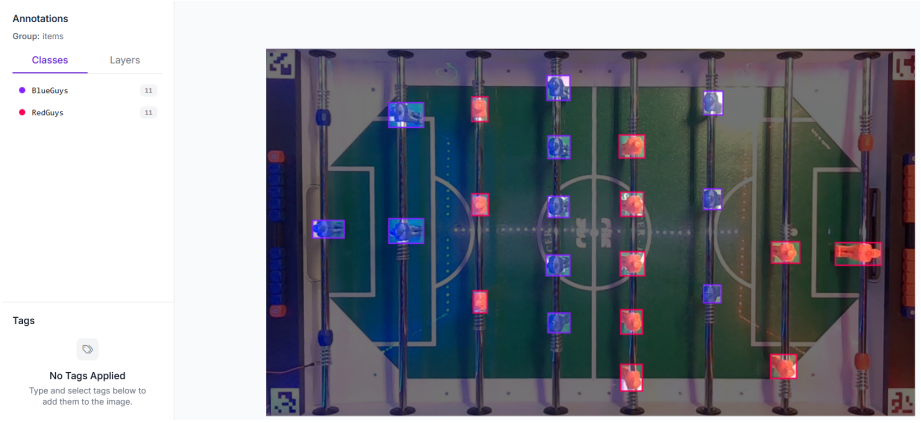

Figure 6.1: Example Annotated Image

Chapter 7

Testing

In this section, we detail how we tested our system by choosing to focus on user testing in our project. This is because our system's reliance on the physical foosball table made traditional unit testing approaches impractical.

7.1 Test Plan

7.1.1 Integration Testing

We chose to do integration testing on the LLM and TTS, since these are very closely connected in the system. When we first integrated both of these components, we had to connect the two in order for the TTS to receive the LLM's messages. We tested this by running the system and playing the game for very specific scenarios. This helped us ensure that we know what we expect as output and to see what we are getting.

7.1.2 User Testing

When we already had the components of the system integrated, we switched to user testing. We played on the table for a while, tried to figure out where and when certain commentary lines could be improved, and then changed the prompts we sent to the LLM accordingly. Furthermore, we asked some of the users of the Foosball table for possible improvements. They told us some crucial rules that they play with and how those should be implemented. The PhD students near the IoT Cyberlab, for example, like to play on the foosball table in their lunch breaks, they are experienced with the table. We asked them to play a couple of games on the table, to see the AI commentator in live action. We noticed that the commentator was able to keep up with the match, provided accurate and engaging commentary without a noticeable delay, and actually made the human players laugh due to funny comments. That they laughed at some of the commentary gave us a proud feeling, since the commentator is intended to bring a competitive, yet fun atmosphere.

Chapter 8

Evaluation

In the following section we evaluate the performance of our final product, and underline the possible improvements that we could have made.

8.1 Performance

Here we review the requirements we did and did not implement, and to what extent the requirement is met. To make it more visually clear, the bullet points that are solid black were implemented, and the white ones were not implemented.

8.1.1 Must Have

- Have a delay of less than 2 seconds between an event (e.g. a goal) and commentary about this event.
- Have the commentator speak understandable English.
- Have the commentator speak with a thrill and intonation.
- Have some commentary at least every 2 seconds, it should not be silent for more than 2 seconds.

  We discovered that having the commentator speak every 2 seconds was quite fast and made it repeat itself too often. This is why we changed this requirement to 3 seconds.
- Commentate on goals, possession, overtake/ball switches teams, serves, and out-of-plays correctly.

8.1.2 Should Have

- Have one of our/staff voices for the commentator.

  Because of time constraints, we did not have the time to record enough voice lines or train a new model.

- Recognize more moves to commentate on.

- Recognize higher level techniques to commentate on.

  Quickly into the implementation we discovered that this requirement was not feasible, due to the design of the code. Currently the code is only given a single frame per iteration. This makes recognizing higher level techniques impossible without first giving multiple previous frames to the detection ai, or creating an advanced prediction algorithm.

- Commentate on assists, dead balls, and faults correctly.

  After discussing with the target audience of the table, most of them just wanted to play casually. For this reason we decided not to implement the more specific faults, because these are not known or used in casual play. We also did no implement commenting on dead balls, because the speed of a Foosball game is so quick that usually before the commentator can commentate on a dead ball, it is already back in play.

8.1.3 Could Have

- Have an actual commentator's voice.

- Recognize situations that are almost happening to comment on.

  We discovered that due to the speed of a foosball game, compared to a regular football game, prediction commentary is not useful, as the predicted events would already happen while the commentator is speaking.

- Commentate on the previous commentary whenever something similar happens ¡- Could be useful if the 2-second silence is approaching.

- Provide users the option to select a team on the web application, for example, the blue team could choose to play with Manchester United players.

- Have current commentating halt in case a more important event happened; if it is commentating on a pass, have it halt to commentate on a goal.

8.2 Possible Improvements

Here we will explain where there are possible improvements to our implementation of the system requirements.

Detection

Have a better camera with higher frame rate, to allow for detection of faster shots. Having this allows for the speed of the shot to be given to the LLM allowing for more accurate commentary.

Have technique detection on multiple frames, or a prediction algorithm to detect higher level techniques. How it is currently applied, the detection is run on a single frame, which makes it very difficult to detect advanced techniques which span multiple frames.

Create a model trained on specific foosball techniques to give to the LLM.

LLM

Have the LLM give more deep insight on the commentary, having it infer situations from the data given, instead of just commenting on the data.

Actual commentators also use what happened previously in their commentary; implementing this can increase the quality of the commentary significantly.

Give the LLM more restricted rules and guidelines, so it gives more accurate commentary. While not specifically mentioned in the requirements the accuracy of the LLM is really important, and it should not make up plays, like that a goal was very fast, while it was slow.

TTS

Currently the TTS sounds pretty good, however it is clearly still robotic. A perfect TTS would be trained on specific foosball commentary and would sound a lot more human.

Chapter 9

Discussion and Conclusion

In this section, we discuss how our team approached this project, as well as our evaluation and improvement points for this approach.

9.1 Planning

In our proposal section we have a table Figure 2.1 showing a rough outline of our planning. Upon starting coding the project, as per our plan, we started with implementing and installing the project code of Foosblitz. However, we realized that the time we had allotted for this was unrealistic. The installation of the code brought a lot of errors and unknowns, the README in the git repository with some instructions, did not explain how to download and install packages and environmental variables. Thus, when downloading the code, we found that we did not use the correct manner of downloading these variables, leading us to more unexpected errors. After a while, we discovered that the environment requires packages and variables to be downloaded through the Poetry[1] environment instead of simply having them on the device. We struggled with this for almost a week, despite help from the Foosblitz project group. By wiping the whole project code and all the packages downloaded with this and re-downloading it all again a couple times, the code ran in the end.

We were hoping that this achievement would allow us to start coding on the AI commentator, but that unfortunately was not the case. When the project was finally able to run, the project would halt; the camera feed would freeze, and the GPUs would stop activity, as soon as something entered the field of vision of the camera, such as a foosball or a hand. We found that we did not have the right CUDA GPU version, which was extremely difficult to resolve within the Pytorch[2] environment. It turned out there was a previous version of Pytorch installed somewhere on the device, so all activities were done through

[1] https://python-poetry.org/docs/managing-environments/
[2] https://pytorch.org/

the old version of Pytorch, instead of the new one in the project code. With assistance from our supervisors we deleted that previous version, and finally got the program to run without breaking upon entry in the field of camera vision.

Setting up the code for the Foosblitz to work properly took longer than anticipated, but we were expecting to make bigger strides now that the foundation worked properly. Unfortunately, we had one more barrier to overcome before we could start on our project. The 'last kicker' detection did not work as it was supposed to, the color detection they had implemented was not functional to the extent we required. In Section 6.2.3 we explain how we implemented the last kicker detection ourselves.

Due to all of these unforeseen circumstances, our planning chart in Figure 2.1 was not useful anymore. Due to this, it becomes difficult to reflect upon our planning, as we no longer have one. We worked by priorities, realizing that to deliver an actual product, we needed at least a minimal functioning AI commentator. So, we focused on implementing our TTS and LLM and combining them into an AI commentator.

9.2 Task Division

Early in the process, we realized we all had different strengths and interests, so we tried to divide our tasks accordingly. From the start we had divided the group into two teams. Szabolcs and Pepijn would work on the TTS, and Daniel, Elena, and Noor would work on the LLM. However, as mentioned in our planning reflection, we quickly shifted tasks to keep up with debugging and installation. Outside of the design process, within the project, the group also needed to write a reflective report. Therefore, Szabolcs, Daniel, and Pepijn stayed focused on the installation of the project code, and Elena and Noor shifted focus to the reflection report. Once over the first hurdle, upon noticing the object detection needed to be implemented differently, Pepijn concentrated on this. Szabolcs and Daniel, after getting the code running, concentrated on the LLM and TTS. Meanwhile, Elena and Noor started tackling the design report, creating the biweekly peer review presentations, and creating the final report presentation for the supervisors and chair.

9.3 Team Evaluation

Prior to this project, we were acquaintances but had not worked in close quarters with everyone yet. We decided to form a group as we all had the same motivation and end goal in mind. We decided to meet most of the week in the IoT Cyberlab to work on the project. Whenever someone was late or absent, this was communicated to every member via our group chat - this way we were all in the loop. We kept each other up-to-date with what tasks we were currently

working on, and since we worked in the same location, whenever we had a bug or an issue, the group was readily available to brainstorm on resolving this.

9.4 Final Results

Our end result met all our must-have requirements, as seen in Section 8.1. Therefore, our project met our expectations and our supervisor's expectations, which we confirmed at our final presentation with our supervisors, clients, and target audience. Our process of meeting with the clients every Wednesday enabled us to use their feedback efficiently, allowing our project to closely align with their needs and wants.

9.5 Improvement Points

As a possible improvement point we would like to mention a Trello[3] board, to keep a better overview of current, completed, and future tasks. After starting with implementing our planning got nullified as debugging took up the whole base of our project, an improvement there could have been to create a new planning to set new expectations for the group. In a more general sense, it could have been nicer to meet earlier most days - we were generally meeting around 13:45 except on Wednesdays (since we had a supervisor meeting at 10:00 every Wednesday).

[3] https://trello.com/home

Chapter 10

Future Work

Our evaluation in Section 8.2 outlines areas where certain requirements could have been implemented more effectively or comprehensively. In this section, we explore potential directions for future development and expansion of our project.

One promising idea that emerged from conversations with our target audience was to enhance the user experience by playing the sound of a cheering audience whenever a goal is scored. Another possible addition would be to have the commentator provide post-match statistics for each plastic player — such as ball possession percentages and other relevant metrics. Similarly, it could be exciting to have statistics on how the Foosball players themselves played. Additionally, offering personalized commentary styles (e.g., serious, humorous, or dramatic) that users can select before the match could make the experience more engaging and tailored to individual preferences. Finally, integrating with streaming platforms would allow remote spectators to watch the match and hear the live commentary, expanding the reach of the system and enabling a shared viewing experience.

Bibliography

[1] Announcing roboflow train 3.0. https://blog.roboflow.com/roboflow-train-3-0/, https://docs.roboflow.com/train/train?ref=blog.roboflow.com. Explanation of the Roboflow 3.0 Object Detection Model, in adition is a link indicating that the Roboflow 3.0 model is based on YOLOv8.

[2] Official Foosball Rules - Foosball Tournament Rules. https://www.foosballsoccer.com/official-foosball-rules.html.

[3] Roboflow: Computer vision tools for developers and enterprises. https://app.roboflow.com/tafelvoetbaltafelballen/foosbetterer/5. Used by the previous group to create a detection model for the ball.

[4] Guide to Official Foosball Rules & How to Play. https://www.imperialusa.com/post/complete-guide-on-the-official-foosball-rules, Jan. 2024.

[5] Czaplicki, M. Live commentary in a football video game generated by an ai. B.S. thesis, University of Twente, 2023.

[6] Prunty, J. Foosball Defense: 8 Secrets You Should Know. https://recroompick.com/foosball-defense/, Jan. 2025.

[7] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016).

[8] ten Klooster, I., Huijskes, H., Vogelezang, M., Ploeg, M., and Takken, S. Design project smart foosball table. *University of Twente* (2024).

Appendices

Appendix A

Prompts

A.1 System Prompt

```
You are an experienced foosball commentator providing live
commentary for a match between Team Blue and
Team Red. Your task is to deliver concise,
engaging, and accurate commentary based on the match
information provided.

For each update which is either an event
or analysis, you will receive match information in
the following format:

<match_info>
<score_blue>{{SCOREBLUE}}</score_blue>
<score_red>{{SCORERED}}</score_red>
<event_description>{{EVENT}}</event_description>
</match_info>

For a filler commentary, you will receive
match information in the following format:

<match_info>
<score_blue>{{SCOREBLUE}}</score_blue>
<score_red>{{SCORERED}}</score_red>
<possession_blue>{{POSSESSIONBLUE}}</possession_blue>
<possession_red>{{POSSESSIONRED}}</possession_red>
<last_kicker>{{LASTKICKER}}</last_kicker>
<previous_last_kicker>{{PREVIOUSLASTKICKER}}</previous_last_kicker>
<ball_in_play>{{BALLINPLAY}}<ball_in_play>
```

```
<event_description>{{EVENT}}</event_description>
</match_info>

Commentary Guidelines:
1. Provide a maximum of 10 words per update.
2. Focus solely on the current action or event described in the input.
3. Use only information explicitly provided in the match info;
do not add external knowledge or speculate.
4. Ensure your commentary sounds natural and engaging.
5. Use short, impactful sentences to maintain audience interest.
6. Avoid repeating phrases or information from previous updates.
7. Stay in character as a professional foosball commentator at all times.

Consider the following:

1. If an event description is provided:
- Identify the key action and who performed it.
- Determine the potential significance of the action in the
current match context.
- Consider the emotional impact of the event on players and audience.
- Plan how to describe this concisely and engagingly.
- Think about potential sound effects or vocal inflections
to enhance the commentary.

2. If no event description is provided (analysis):
- Note the current score and consider its implications.
- Calculate the possession difference and its potential impact.
- Analyze the significance of the current game state.
- Consider how this might affect game strategy.
- Think about the overall mood or tension of the match at this point.

3. For filler commentary (e.g., predictions):
- Avoid welcome messages or introductions.
- Focus on unique aspects of the current game situation.
- Choose different variables from the match_info in each update
so the commentary is unique every time.
- Commentate on the last kicker, previous last kicker and
possession equally

You might also receive information about zones on the football table,
these are the names of the zones and what they correspond to:
0: Blue goalkeeper
1: Blue defenders
2: Red attackers
3: Blue midfielders
4: Red midfielders
```

```
5: Blue attackers
6: Red defenders
7: Red goalkeeper
-1: Out of play

Provide your final output in either <commentary> or <analysis> tags,
depending on the input type.

Example output structure:

<commentary>
[Concise, engaging description of the current action.
Brief comment on its significance.]
</commentary>

or

<analysis>
[Brief analysis of the current game situation
based on score, time, and possession]
</analysis>

or

<filler>
[Brief filler commentary using the provided
variables from the match_info]
</filler>

Remember, your final output should not exceed 8 words
and should be highly engaging and informative.

Please provide your commentary or analysis
for the given match information.
```

A.2 Event Prompt

```
<match_info>{game_info_string}<event_description>{event}
</event_description></match_info>
```

A.3 Filler Prompt

No commentary was done for a while. Create unique filler commentary to keep the viewers engaged. Do not repeat previous filler commentary. Choose randomly only one of the given data to make filler commentary.