I'll take a bit about our challenges and about the algorithms we've used for controling the state of the game.

1 Evil / Good Hangman Algorithm

As Fran just said, we wanted to control the outcomes of the game. This was quite a challenge, since even if you use a difficult word, some users might still be able to figure it out. For this reason, we've developed an algorithm that's based on one of Stanford's nifty assignments. The algorithm for evil hangman works as follows:

First of all, I'll present you, how we made the Nao evil, and then how we made it good (the algorithms work very similar).

Okay, so the idea behind evil hangman is, that the Nao dodges the guesses of the user. Every time a user guesses a letter, it avoids this letter. The evil thing is that the Nao does not decide for a word until it has to do so.

1. Step 1: The Nao decides for a certain word length, for example four letters but not for a word (this is the first step of being evil).
2. Step 2: Then the user makes a guess, for example E.
3. Step 3: And now, the Nao gets a bit more evil. It uses its processing power for creating equivalence classes or word families. All word in an equivalence class have the same shape, when displayed.
4. Step 4: And now the real evil thing happens. The Nao choses the largest equivalence class that does not contain the guessed letter. item
5. Step 5: Then, it uses this equivalence class as new dictionary and waits for the next input from the user.

And now for good hangman. Anyone has an idea, how we've implemented that? Yeah, we've just used the largest word family that *contains* the letter.

2 App

Okay, as said before, our next challenge was how to display the game status. We thought about the Nao that draws on a white-board but you might imagine that we've dodged this idea quite quickly. Next, we've thought that the user should keep track but that's somehow annoying and cumbersome. So we've developed an app. The app is written in Kivy, Kivy is a cross-platform python framework and allows for developing apps for Linux, Windows, iOS, and Android. The app looks like that (show tablet and mobile phone). AS you might imagine, developing the app was quite time-consuming, since there are a lot of little things that you have to think of.

3 Connection Nao and App

Okay, then, we had the app, however the question was, how can the Nao talk to the app and the app to the now?

Then we came up with the client-server model. The Nao and the app are front-ends and they are communicating via a server. So the Nao is sending post requests with data, the server stores this data, and the app is querying the database.

This has several advantages, for example the game status can be displayed simultaneously on several devices. This allows the experimenter to keep track of